Proceedings of the 2007 IEEE Symposium on Computational
Intelligence in Image and Signal Processing (CIISP 2007)

Enhanced Shot-Based Video Adaptation using MPEG-21 generic Bitstream Syntax Schema

Sarah De Bruyne, Davy De Schrijver, Wesley De Neve, Davy Van Deursen, 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: {sarah.debruyne, davy.deschrijver, wesley.deneve, davy.vandeursen, rik.vandewalle}@ugent.be

Abstract— **Semantic video adaptation takes into account the relevance of the different fragments of the video content in order to create a tailored video stream based on the user's preferences. As a shot can be considered as the smallest semantic unit in a video sequence, metadata can be added to each shot using MPEG-7 descriptions. Based on these metadata and the user's preferences, the original bitstream can be adapted in order to obtain the desired fragments. MPEG-21 DIA offers a tool, gBS Schema, for exposing the high-level structure of a binary resource as an XML description. In this paper, shot information is inserted in these descriptions to create a link between metadata and semantic video adaptation. Furthermore, this paper proposes to keep the structure of these descriptions format-agnostic. As a result, only one generic transformation style sheet has to be implemented to support shot-based video adaptation of sequences compliant with different video specifications. Special attention is payed to sequences coded with the H.264/AVC standard as this specification contains several interesting features important for shot-based video adaptation.**

I. Introduction

As multimedia has proliferated over the past years, many new technologies have been developed to establish the delivery and consumption of multimedia content. Users began to expect that this content can easily be accessed according to their own preferences. Therefore, the delivered content must be tailored to the user's characteristics and preferences, as well as to the capacities of the terminals and networks.

Video adaptation [1] is an emerging field of interest that includes techniques responding to the above challenges. Several adaptation strategies can be identified, either operating on a semantic level (e.g., removal of violent scenes or extraction of semantic highlights), at a structural level (e.g., key frame extraction), or at signal-processing level (e.g., transcoding).

To adapt a video sequence, MPEG-21 Digital Item Adaptation (DIA) [2] offers a tool, generic Bitstream Syntax Schema (gBS Schema), to describe the high-level structure of the bitstream using the Extensible Markup Language (XML). The resulting XML document is called a generic Bitstream Syntax Description (gBSD) which makes it possible to describe the bitstream in a coding format-agnostic manner.

This paper concentrates on the link between metadata and format-agnostic semantic video adaptation by making use of gBS Schema. This way, metadata and semantic video adaptation can be coupled in an elegant manner. Therefore, shot information is inserted in the gBSDs indicating to which shot each frame belongs. The selection of the desired shots can be obtained by using MPEG-7 descriptions containing metadata about the different shots. Once the desired shots are indicated, a generic transformation style sheet is used to obtain the desired adapted sequence by linking the desired shots to the shot information available in the gBSD. Special attention needs to be payed to the extraction of the desired fragments as the adapted bitstream needs to remain compliant with the corresponding specification.

Related work includes a semantic adaptation framework for the generation of semantic metadata and the semantic adaptation of video on a frame basis using gBS Schema [3]. Furhtermore, [4] and [5] focus on video adaptation using gBS Schema. In particular, an example of a gBSD is given which is used to classify fragments of a video using semantic information.

This paper is organized as follows. The following section introduces the main enabling technologies and concepts, while Sect. III discusses the shot-based adaptation process. Experimental results are given in Sect. IV.

II. Enabling Technologies and Concepts

A. gBSD-driven Content Adaptation

MPEG-21 gBS Schema is a tool of part 7 (Digital Item Adaptation, DIA) of the MPEG-21 specification used to facilitate content adaptation [4], [5]. To realize this, gBS Schema defines a framework that enables the description of the high-level structure of a bitstream in XML, resulting in a Bitstream Syntax Description (BSDs). This description is not meant to describe the bitstream on a bit-per-bit basis, but rather addresses its high-level structure. In Fig. 1, a global architecture for a BSD-based content adaptation framework is given. First, a BSD of the high-level structure of the bitstream is generated. This BSD is then adapted according to the user's preferences by means of a transformation language. Finally, the adapted BSD becomes input to an adaptation module responsible for the generation the corresponding bitstream.

gBS Schema uses only one generic schema to describe the structure of a generic BSD (gBSD), making the syntax of the gBSD generic and codec-independent. Therefore, the regeneration of the adapted bitstream can be achieved without the need of codec-specific schemas. Furthermore, this schema makes it possible to describe the bitstream in a hierarchical fashion and provides semantically meaningful marking of syntactical

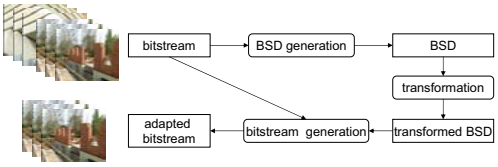

Fig. 1. Architecture for a BSD-based content adaptation framework

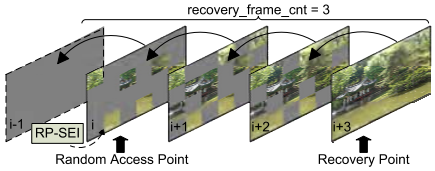

Fig. 2. Gradual random access applied to H.264/AVC

elements by the use of a "marker" handle facilitating semantic-based adaptations.

Since the gBS Schema specification can be found in the MPEG-21 DIA standard [2], only a brief summary of the most important elements needed within the scope of this paper is given. Examples of gBSDs illustrating the described concepts are given in Sect. III.

- The `gBSDUnit` element represents a bitstream segment by referring to the corresponding location in the bitstream. Each gBSDUnit can then further consist of gBSDUnits and Parameters resulting in a hierarchical representation of the bitstream. A gBSDUnit includes a *start* and *length* attribute to point to the section in the bitstream it describes. In addition, it can also contain a *syntacticalLabel* attribute for including coding-format specific information about the hierarchical structure identified via classification scheme terms while the optional *marker* attribute provides semantic information used for performing adaptations.
- The `Parameter` element is used to describe a syntax element in the bitstream of which the value might be changed during the adaptation process. Therefore, it provides the actual value and the data type of the corresponding bitstream fragment. Similar to the gBSDUnit, it can also contain a *syntacticalLabel* and a *marker*.

B. Random Access in Video Coding

In video coding, improved compression efficiency is achieved by taking advantage of the large amount of temporal redundancy in video content. However, temporal prediction causes inconveniences in other aspects such as random access.

As discussed by Hannuksela et al. in [6], random access refers to the ability of the decoder to start decoding at a point in a video sequence other than at the beginning and to recover an exact or approximate representation of the decoded pictures. This random access operation is characterized by a random access point and a recovery point, as can be seen in Fig. 2. The random access point appears first and is a coded picture where the decoding can be initialized. The recovery point indicates that the content of all decoded pictures at and subsequent to this frame is correct or approximately correct.

The random access operation is called Instantaneous Decoding Refresh (IDR) when the random access point and the recovery point coincide. As a consequence, the corresponding frame will be intra coded. In case the random access point and the recovery point do not concur, frames in between the two points will contain artifacts and the random access process will be gradual (Fig. 2). The latter process is called Gradual Decoding Refresh (GDR) and refers to the ability to start decoding at a non-IDR picture and to recover decoded pictures that are correct after decoding a certain amount of pictures.

Depending on the application area, random access points are mostly inserted in video sequences on a regular time basis or based on the video contents. The first case is mostly used in streaming applications such as broadcasting whereas the second case is more often used in applications where high compression ratios are preferred. Consequently, in the last case, random access points often coincide with shot boundaries because the content of the current frame will highly differ from the previous frames belonging to the previous shot.

During most semantic adaptation processes, the extraction of certain segments is desired. In case the beginning of a segment corresponds to a random access point, the extracted video can be decoded without any problem. However, in the other case, special precautions need to be taken in order to extract the desired segment as described in Sect. III.

C. Random Access Applied to H.264/AVC

In earlier video specifications, each intra-coded picture corresponds to a random access point as subsequent frames are not allowed to refer to pictures located before this intra-coded picture. In H.264/AVC [7], this principle does no longer apply because of the introduction of the multiple reference picture buffer. Therefore, intra-coded pictures which correspond to a random access point are explicitly marked as IDR pictures.

To indicate gradual random refresh, H.264/AVC provides Recovery Point Supplemental Enhancement Information messages (RP-SEI message) as can be seen in Fig. 2. The frame associated with an RP-SEI message corresponds to a random access point. To signal the corresponding recovery point, this message contains a `recovery_frame_cnt` element which indicates the number of reference frames that need to be decoded to arrive to the recovery point.

III. Shot-Based Adaptation using gBS Schema

The idea behind semantic video adaptation is the extraction of the desired fragments based on the user's preferences. This can be done by using MPEG-7 descriptions containing metadata about the content of the video. Based on these metadata, it is possible to locate the desired parts. For example, two different people would like to see an overview of a football match. The first person is only interested in the fragments containing goals, while the second person would like to see all fragments of his favorite player. By inserting these keywords

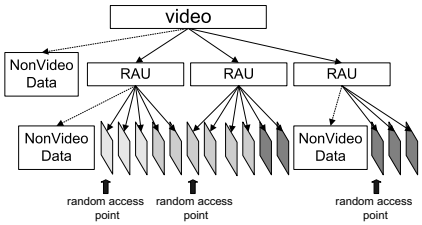

Fig. 3. Example of the structure of a video containing multiple shots indicated by different grey tints

```
<dia:DIA>
 <dia:DescriptionMetadata>
  <dia:ClassificationSchemeAlias alias="VC" href="urn:generalVideoCoding:
        syntacticalLabels"/>
 </dia:DescriptionMetadata>
 <dia:Description xsi:type="gBSDType" addressUnit="byte" addressMode="Absolute" bs1
       :bitstreamURI="c:\video_sequence.264">
  <gBSDUnit syntacticalLabel=":VC:NonVideoData" start="0" length="8" marker="
        necessary"/>
  <gBSDUnit syntacticalLabel=":VC:NonVideoData" start="8" length="13" marker="
        necessary"/>
  <gBSDUnit syntacticalLabel=":VC:RAU" start="21" marker="Shot_0">
   <gBSDUnit syntacticalLabel=":VC:Frame" start="21" length="272" marker="Shot_0"/>
   <gBSDUnit syntacticalLabel=":VC:Frame" start="293" length="94" marker="Shot_0"/>
   <!-- successive frames belonging to this RAU -->
  </gBSDUnit>
  <!-- other RAUs -->
  <gBSDUnit syntacticalLabel=":VC:RAU" start="740" marker="Shot_2 Shot_3">
   <gBSDUnit syntacticalLabel=":VC:Frame" start="740" length="13" marker="Shot_2"/>
   <gBSDUnit syntacticalLabel=":VC:Frame" start="753" length="55" marker="Shot_3"/>
   <gBSDUnit syntacticalLabel=":VC:Frame" start="808" length="84" marker="Shot_2"/>
   <gBSDUnit syntacticalLabel=":VC:Frame" start="892" length="92" marker="Shot_3"/>
   <!-- successive frames belonging to this RAU -->
  </gBSDUnit>
  <!-- other RAUs -->
 </dia:Description>
</dia:DIA>
```

Fig. 4. Fragment of a gBSD used for semantic adaptation

in an MPEG-7 description, it should be possible to meet everyone's needs. As a shot is considered to be the smallest semantic unit in a video sequence, keywords are added to each shot in this MPEG-7 description. In case a person would like to see all goals, the adaptation engine selects all shots containing this keyword. By inserting shot information in the gBSDs corresponding to the video sequences, a link can be made between the metadata and the adaptation.

In the following sections, we will describe how this shot-based adaptation process can be achieved using gBS Schema.

A. Structure of the gBSD

Frames belonging to the same shot are often grouped in one gBSDUnit so that the extraction of the desired shot can easily be achieved [5]. However, this approach causes two problems. The first problem arises when the starting frame of a shot in decoding order does not coincide with a random access point. Consequently, all frames located between the previous random access point and the starting frame need to be added to the adapted bitstream as well, in order to make correct decoding possible. A second problem is attributed to the difference between display order and decoding order. Let $P_{a1}B_{a2}P_{b1}B_{b2}P_{b3}$ be a video sequence in display order containing two shots a and b, the corresponding decoding order could then be $P_{a1}P_{b1}B_{a2}P_{b3}B_{b2}$. One can see that the frames belonging to a shot do no longer succeed each other uninterruptedly. Since a gBSD describes the structure of the coded bitstream in decoding order, it will be impossible to group frames belonging to the same shot together in a gBSD.

To resolve these problems, we propose a new hierarchical structure for gBSDs in the context of shot-based adaptation. Instead of dividing a video sequence into shots, we will group frames belonging to the same "Random Access Unit" (RAU) together, as can be seen in Fig. 3. This RAU contains a number of successive frames in decode order, starting with a frame corresponding to a random access point and ending just before the next random access point. This RAU is represented by a gBSDUnit and is further divided in gBSDUnits representing the frames belonging to the RAU (Fig. 4).

As frames belonging to one shot are no longer gathered, this information needs to be stored for every frame separately. The *marker* attribute offers a good solution to store shot information as it is intended to provide semantic information. Furthermore, for each RAU, a *marker* is appended, giving a survey of the shot information included in the RAU. This extra information makes it possible to speed up the adaptation process. In order to extract a particular shot, it is no longer necessary to take all frames into consideration. Instead, by examining the markers of RAUs, only RAUs containing a reference to the wanted shot need further exploration.

In contrast to *marker* attributes, the *syntacticalLabel* attribute is used to include specific information about the hierarchical structure of the bitstream. In most applications, these names are codec-specific, thus enabling codec-aware adaptations. However, in our application, we prefer to use a general structure making it possible to implement a format-independent semantic adaptation style sheet. In this case, it is possible to employ only one transformation style sheet that is able to semantically adapt bitstreams compliant with different coding specifications.

Besides frames, most video specifications insert additional information into a video bitstream. In the H.264/AVC standard, for example, non-VCL (non-video coding layer) NAL units are used to insert additional information, such as parameter sets and SEI messages into the bitstream. As most video specifications use different techniques and syntax elements to insert non-VCL information, the insertion of a format-specific information would lead to a format-specific gBSD. As this is undesired, only one format-agnostic element "NonVideoData" is added. To make a difference between NonVideoData applicable to the whole video sequence and information applicable to only one shot, extra information is added to the corresponding marker in the gBSD. In Fig. 4, the first two gBSDUnits, corresponding to certain parameter sets, are marked as necessary because these units could be referred to by all frames in the sequence. In Fig. 5 on the other hand, some "NonVideoData" parameters are marked according to the shot they belong to. More explanation about these shot-specific parameters is given below.

B. Concealment of Undesired Fragments

In a number of applications, it is desired that fragments not belonging to the specified shot(s) are not shown. This problem

arises when the first frame of the shot does not coincide with the corresponding random access point. Therefore, the undesired frames located between the random access point and the first frame of the shot need to be conceiled.

A way to cope with this problem is by making use of features of the MP4 file format. This container makes it possible to create a mapping between the different frames and their display time, making it possible to conceal undesired frames. However, a drawback of this approach is the fact that the video bitstream as well as the MP4 file need to be adjusted.

A number of video specifications contain special features to overcome this problem. In H.264/AVC, it is possible to specify an own-defined SEI message indicating the frames not permitted to be shown. Since this message is not included in the specification, decoders are not able to interpret this message and will therefore discard it. A better solution is to use messages already available in the specification.

In H.264/AVC, RP-SEI messages are used for gradual random access. As already elaborated in Sect. II, this message indicates the position of the random access point and the corresponding recovery point. Frames located in between these two positions are considered incorrect and are therefore not displayed by a decoder. By inserting an RP-SEI message in the beginning of a RAU, undesired fragments can be concealed as well, although their content does not contain artifacts.

Fig. 5 gives an example of an RAU containing a "FrameConcealment" gBSDUnit which represents an RP-SEI message. This unit will be inserted in the adapted bitstream in case it contains information about a desired shot, as indicated by the marker. The FrameConcealment unit consists of two major types of information.

- The first type is marked as necessary and is present in all RP-SEI messages. This information includes for example the starting bytes, the NAL unit type, and the SEI message type of the NAL unit.
- The second type is different for each shot and contains information like the offset of the recovery point. The marker indicates to which shot this information belongs.

In the example, the RAU contains frames belonging to two shots, i.e., *shot_0* and *shot_1*. In case the adaptation engine decides to extract shot_1, we expect that the frames located before the first frame of shot_1 are not displayed. Therefore, the FrameConcealment unit will be inserted in the adapted gBSD as its marker contains a reference to shot_1. However, not the whole FrameConcealment unit will be included, but only those parts marked as necessary for shot_1. As a result, the adapted bitstream will contain an RP-SEI message indicating that the frames located before the recovery point (which coincides with the first frame of a shot) need to be decoded but are not allowed to be displayed.

A restriction that needs to be kept in mind is the fact that the element indicating the recovery point, i.e., `recovery_frame_cnt`, needs to be in the range of 0 to *MaxFrameNum-1*. As the offset of the first frame of a shot can be higher than this limit, a solution needs to be found. A first possible solution is to weaken this constraint by extending its range, but this is undesired as the adapted bitstream will no longer be compliant with the H.264/AVC specification. Another possibility is to modify the element responsible for *MaxFrameNum*. However, this would influence all frames in the sequence, which makes this approach unfeasible. The best solution is to insert additional RP-SEI messages just before the previous recovery point until the starting frame is reached. Even though this approach leads to some overhead, we consider it to be the most elegant as the adapted bitstream remains compliant with the H.264/AVC specification and the amount of changes made to the bitstream is minimal.

```
<gBSDUnit syntacticalLabel=":VC:RAU" start="62476" length="16945" marker="Shot_0
      Shot_1">
  <gBSDUnit syntacticalLabel=":VC:FrameConcealment" start="0" marker="Shot_0 Shot_1
      ">
    <gBSDUnit syntacticalLabel=":VC:NonVideoData" start="0" marker="necessary">
      <!-- information like start of the NAL unit, NAL unit type, SEI type, ...-->
    </gBSDUnit>
    <gBSDUnit syntacticalLabel=":VC:NonVideoData" start="0" marker="necessary">
      <Parameter name=":VC:NonVideoData" length="2" marker="Shot_0">
        <Value xsi:type="xsd:hexBinary">396</Value>
      </Parameter>
      <Parameter name=":VC:NonVideoData" length="3" marker="Shot_1">
        <Value xsi:type="xsd:hexBinary">135728</Value>
      </Parameter>
    </gBSDUnit>
    <Parameter name=":VC:NonVideoData" length="1" marker="necessary">
      <Value xsi:type="xsd:hexBinary">128</Value>
    </Parameter>
  </gBSDUnit>
  <gBSDUnit syntacticalLabel=":VC:Frame" start="476" length="152" marker="Shot_0"/>
  <!-- successive frames belonging to this RAU -->
  <gBSDUnit syntacticalLabel=":VC:Frame" start="843" length="133" marker="Shot_1"/>
  <!-- successive frames belonging to this RAU -->
</gBSDUnit>
```

Fig. 5. Fragment of a FrameConcealment gBSDUnit

C. gBSD Generation

As the gBS Schema is generic, it is not possible to automatically generate gBSDs using this schema. A possibility is to use a format-specific schema to generate BSDs, which can subsequently be transformed into the corresponding gBSDs. [5]. However, as our adaptation process is shot-based, the shot boundaries need to be detected as well. Since the entire structure has to be analyzed during the shot detection [8], we prefer to generate the gBSDs during this analysis process.

D. Shot-Based Adaptation of the gBSD

To adapt the video to the user's preferences, only the desired fragments need to be extracted. As the high-level structure of the video is described in XML, the extraction process can be done by using Extensible Stylesheet Language Transformations (XSLT). Fig. 6 depicts the data flow described by the XSLT style sheet responsible for our shot-based adaptation process. To indicate which fragments are desired, the parameter `wanted_shots`, containing a list of the desired shots, is passed to the style sheet. Afterwards, this style sheet transforms the gBSD by making use of templates working on *Units*, corresponding to *gBSDUnits* as well as *Parameters*.

For each *Unit* present in the top level of the description, i.e., RAUs and NonVideoData (Fig. 3), the template *Process Unit* is called. NonVideoData *Units* will be added to the bitstream in

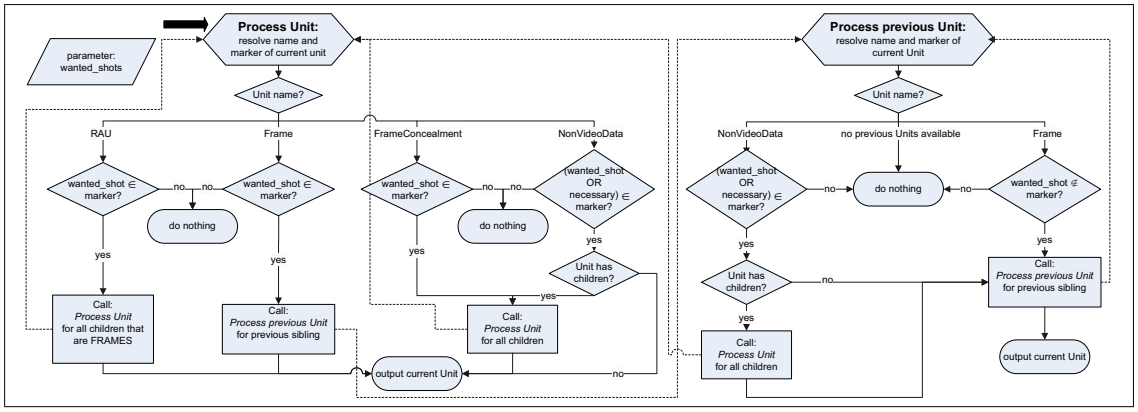

Fig. 6. Data flow described by the XSLT style sheet responsible for our shot-based adaptation process

case the unit is marked as necessary or when a shot, contained in wanted_shots, corresponds to an element of the marker. As some NonVideoData *Units* further contain NonVideoData *Units*, their children will be processed on their turn as well.

The RAUs are further processed only in case a shot contained in wanted_shots corresponds to an element of the marker; otherwise, the RAU is discarded and the algorithm directly proceeds with the next *Unit*. In the former case, the RAU will contain frames belonging to a desired fragment, and therefore, certain frames present in the RAU will need to be added to the adapted bitstream. As a result, all frames present in the RAU need to be processed by calling *Process Unit*.

Since it is necessary to start a video sequence at a random access point, *Units* located before the frames belonging to the desired shot need to be added to the adapted gBSD as well. As XSLT has no such construct as a "while" construct, recursion is needed. Consequently, *Process previous Unit* is called for each frame belonging to a desired shot This unit will then be checked to see whether it needs to be added to the transformed gBSD or not. If so, the recursion is repeated until the adapted bitstream can be decoded. The recursion terminates when the beginning of the RAU is reached or when the processed unit is already added to the bitstream. To conceal certain frames, FrameConceilment *Units* are added in case their marker contains one of the desired shots.

As no codec-specific information is used to transform the gBSD, this transformation can be executed on gBSDs corresponding to sequences compliant with different specifications.

E. Bitstream Reconstruction

The gBSDtoBin process is normatively specified in the MPEG-21 DIA specification. The gBSDtoBin parser generates a bitstream by using the information available in the transformed gBSD. To reconstruct the bitstream, segments referred to by gBSDUnits are copied to the resulting bitstream while the values of Parameters are inserted in the bitstream, hereby taking into account the corresponding data type.

IV. Performance Results

To evaluate the performance of our shot-based adaptation process, several experiments have been performed on a video sequence coded several time using different parameter settings. Time measurements of the different adaptation steps were carried out and the sizes of the gBSDs and the corresponding bitstreams were compared.

A. Methodology

Experiments have been carried out on the trailer of "Friends with money" containing 2353 frames and 49 shots. This sequence was coded several times with variable as well as with fixed (IB(PB)*) GOP structures. For the fixed GOP structures, IDR frames were inserted every 10, 100 and 200 reference frames respectively. For the variable GOP structure, the location of the random access point depends on the content of the video. These bitstreams were coded multiple times with different values for *maxFrameNum*. For all values of *maxFrameNum* higher than the size of a GOP, maximum one RP-SEI message needed to be inserted in an RAU. Therefore, the results of only one bitstream with *maxFrameNum* higher than the size of a GOP are presented. As the encoder chooses the optimal value for *maxFrameNum* when using variable GOP structures, only one result for this GOP structure is presented.

B. Discussion of the Results

The performance results of the different steps are shown in Tables I and II. The first step in the adaptation process is the creation of the gBSDs. Since this step is performed during shot boundary detection, time measurements are not provided as they are not representative. The sizes of the created descriptions, compared to the sizes of the corresponding bitstream, are shown in Table I. As the size of a bitstream hardly increases when a higher *MaxFrameNum* is adopted, only the size of one original bitstream is presented, i.e., the bitstream corresponding to the gBSD size marked in italic.

The experiments show that the size of the gBSD decreases when *maxFrameNum* increases. This can be attributed to the fact that less RP-SEI messages need to be inserted in one

TABLE II

AVERAGE EXECUTION TIME (MS) FOR THE XSLT TRANSFORMATION T AND THE GENERATION G OF THE ADAPTED BITSTREAM REPRESENTING ONE SHOT. THE AVERAGE SIZE S (KB) OF AN ADAPTED BITSTREAM IS GIVEN AS WELL

GOP structure	original bitstream size (KB)	gBSD without RP-SEI			gBSD with RP-SEI based on MaxFrameNum														
		T	G	S	transformation (ms)					generation (ms)					size (KB)				
		(ms)	(ms)	(KB)	16	32	64	128	512	16	32	64	128	512	16	32	64	128	512
variable	4702	674	71	111.3					674					72					111.3
IDR 10	4747	661	75	122.7	700					82					122.7				
IDR 100	3876	699	92	246.7	726	717	708	707		96	99	100	93		226.0	245.9	238.7	246.8	
IDR 200	3839	746	107	357.1	777	765	756	753		106	113	108	107		331.7	357.2	356.7	347.0	

TABLE I

SIZE OF THE GBSDS COMPARED TO THE ORIGINAL BITSTREAM (KB)

GOP structure	original bitstream size	gBSD without RP-SEI	gBSD with RP-SEI MaxFrameNum					
			16	32	64	128	256	512
variable	4702	226						*261*
IDR 10	4747	227	*340*					
IDR 100	3876	216	*309*	271	247	234		
IDR 200	3839	215	*328*	274	251	236		

RAU. The differences in size of the gBSDs containing RP-SEI messages for the various GOP structures is less obvious. gBSDs corresponding to bitstreams with large GOP structures will consist of less RAUs, but will contain more shots and therefore contain more and larger RP-SEI messages. On the other hand, bitstreams coded using small GOPs will contain more RAUs consisting of smaller RP-SEI messages. By compressing the gBSDs, the overhead caused by the descriptions is negligible.

The gBSD transformation is the next step in the adaptation process. The execution times in Table II represent the average of the times needed to generate an transformed gBSD for each shot in the video sequence. As video sequences coded using large GOPs consist of larger RAUs, more shots are grouped together in one RAU. This leads to a longer execution time since more frames need to be processed. Video sequences coded with a variable GOP structure will outperform video sequences coded with fixed GOPs as the division of the RAUs is connected with the content of the video (i.e., the shots).

Finally, the MPEG-21 gBSDtoBin reference software was used to create the adapted bitstream containing the desired shot. From Table II, one can conclude that the generation of the adapted bitstream can be done in real time. Furthermore, Table II also represents the average size of an adapted bitstream corresponding to one shot. Video sequences coded with a variable GOP structure turn out to be the most appropriate for shot-based adaptation as the size of their gBSDs and the corresponding bitstreams is small and the execution times outperform those of the other sequences. For video sequences coded with a fixed GOP structure, sequences consisting of small GOP structures will outperform those with large GOP structures because less concealed frames need to be added. On the other hand, more random access points need to be inserted resulting in a decrease of the compression efficiency.

For other sequences, similar results are obtained.

V. CONCLUSION

This paper introduced an enhanced shot-based adaptation framework bridging the gap between format-agnostic semantic video adaptation and metadata by making use of gBS Schema. As the proposed hierarchical structure of the gBSDs is format-agnostic, only one generic transformation style sheet is needed. By inserting shot information into the descriptions, the adaptation process can be steered in order to extract the desired fragments. During the extraction, special attention is payed to random access so that the bitstream remains compliant with the corresponding specification. Experiments show that the adaptation and regeneration can be done in real time and that video sequences coded with variable GOP structures are more suited for shot-based adaptations than fixed GOP structures.

ACKNOWLEDGMENT

The research activities as 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] S.-F. Chang and A. Vetro, "Video adaptation: Concepts, technologies and open issues," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 148–158, January 2005.

[2] ISO/IEC JTC 1, "Information Technology – Multimedia Framework (MPEG-21) – Part 7: Digital Item Adaptation," *ISO/IEC 21000-7:2004*, October 2004.

[3] M. Zufferey and H. Kosch, "Semantic adaptation of multimedia content," *Proc. of WIAMIS 05*, April 2005.

[4] C. Timmerer, G. Panis, H. Kosch, J. Heuer, H. Hellwagner, and A. Hutter, "Coding format independent multimedia content adaptation using XML," *Proc. of SPIE International symposium (ITCOM 03)*, vol. 5242, no. 3, pp. 92–103, September 2003.

[5] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers, and M. Amielh, "Bitstream syntax description: a tool for multimedia resource adaptation within MPEG-21," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 721–747, September 2003.

[6] M. M. Hannuksela, Y.-K. Wang, and M. Gabbouj, "Isolated regions in video coding," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 259–267, April 2004.

[7] ITU-T and ISO/IEC JTC 1, "ISO/IEC 14496-10:2004 Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," 2004.

[8] S. De Bruyne, W. De Neve, K. De Wolf, D. De Schrijver, P. Verhoeve, and R. Van de Walle, "Temporal video segmentation on H.264/AVC compressed bitstreams," *Lecture Notes in Computer Science - Advances in Multimedia Modeling - MMM 2007, Part I*, vol. 4351, pp. 1–12, 2007.