Exploitation of Interactive Region of Interest Scalability in Scalable Video Coding by Using an XML-driven Adaptation Framework

Davy De Schrijver, Wesley De Neve, Davy Van Deursen, Sarah De Bruyne, 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: davy.deschrijver@ugent.be

Abstract

Interactive Region Of Interest (IROI) scalability is one of the requirements of the Scalable Video Coding extension of H.264/AVC (SVC). IROI can be used in applications in which the most interesting part of the video pane (in particular, the ROI) cannot be defined during the encoding phase. Therefore, the coded bitstreams have to be created in such a manner that the ROI can be selected on-the-fly at any moment. In this paper, we show how these bitstreams can be generated within the SVC specification by relying on the Flexible Macroblock Ordering (FMO) tool. By using this tool, it is possible to divide a frame in independent decodable tiles or slices. We subsequently explain the extraction process to select the slices belonging to the desired ROI. Furthermore, in this paper, we shift the adaptation process (i.e., the extraction of the ROIs) from the binary domain to the XML domain. This gives us the opportunity to rely on a format-agnostic adaptation engine which only makes use of an XML-based bitstream syntax description and a transformation stylesheet. The use of the MPEG-21 Bitstream Syntax Description Language will be discussed to obtain the XML-based bitstream syntax descriptions. The performance of the different steps in this framework is investigated, in particular, the overhead of coding a bitstream once with and once without the IROI scalability, the generation time of the XML descriptions, the extraction process implemented in Streaming Transformations for XML (STX), the regeneration of the adapted bitstream, and finally the quality and bit rate of the different bitstreams.

1. Introduction

In many video-based application scenarios (e.g., surveillance systems and video conferencing), it can be interesting to work with Regions Of Interest (ROIs). These ROIs typically contain the most important or interesting parts of the video pane. The selection of the ROIs is user and application dependent. In most on-line applications, the ROI is defined during the encoding process, resulting in the fact that the ROI is extracted and the quality of the background is decreased.

However, in a number of scenarios, the ROI cannot be pointed out during the encoding process and has to be selected on-the-fly during an adaptation process. Such a scenario can for example appear in surveillance systems. As long as nothing suspicious appears on the screen, the video shall contain a constant quality. At the moment that a questionable situation arises, the guard can select the interesting part of the pane and this region will be transmitted at a high quality, while the surrounding background has a degraded quality. At the same moment, another guard, walking around with a PDA at his disposal, may be interested in another part of the video pane. He can extract another ROI from the same coded bitstream in an on-the-fly fashion. Such an application scenario is shown in Figure 1.

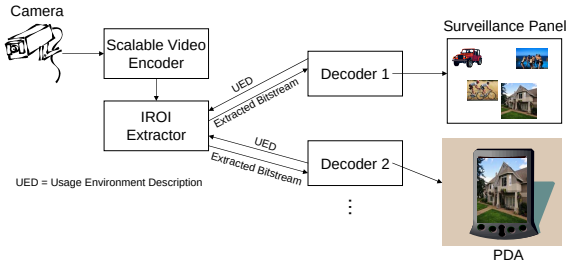

Figure 1. An example of IROI scalability in a surveillance system

In this context, we are talking about Interactive Regions Of Interest (IROI). In JPEG2000, the concept of IROI was already introduced for still images [11]. In this case, the ROI

is not coded during the encoding process but is determined by an extractor. Therefore, the encoder has to generate bitstreams in such a way that the ROI can be selected in an interactive manner. This means that these bitstreams have to be IROI scalable. Therefore, in this paper, we will use the Scalable Video Coding extension of H.264/AVC (further abbreviated as SVC) specification because one of its requirements is as follows: *scalable coding should support a mechanism that permits interactive rectangular region of interest scalability with an access granularity of 32 pixels* [12].

In the first part of this paper, we discuss the technologies used in order to create and to adapt IROI scalable bitstreams. This is particularly done using the Flexible Macroblock Ordering (FMO) tool as standardized in the H.264/AVC specification [2]. Afterward, we explain how the desired ROIs can be extracted from an SVC bitstream. This algorithm, which is only based on syntax elements available in the bitstream, is used to extract the selected ROIs in an interactive way. This is discussed in Section 3.

Furthermore, we want to obtain a format-agnostic content adaptation framework. Therefore, we prefer to realize the extraction process in the XML domain [10]. The MPEG-21 Bitstream Syntax Description Language (MPEG-21 BSDL, [15]) framework can be used to realize the content adaptation process in the XML domain, in particular for exploiting the IROI scalability. In this paper, we describe how the MPEG-21 BSDL framework can be used to obtain XML descriptions of the bitstreams, and how the tailored bitstreams can subsequently be generated by using transformed XML descriptions.

Finally, a number of performance results of the XML-driven framework are discussed in Section 4. In particular, the overhead of coding a bitstream once with and once without the IROI scalability, the generation time of the XML descriptions, the extraction process implemented in streaming transformations for XML, the regeneration of the adapted bitstream, and finally the quality and bit rate of the different bitstreams are investigated. The conclusion of this paper is drawn in Section 5.

2. Fundamental Technologies

2.1. IROI Scalability in H.264/AVC Scalable Video Coding

IROI scalability means that the encoder generates a scalable bitstream without knowledge about the possible ROIs (coding process). Then, the ROI can be selected on-the-fly by the user (selection process). Because there are no predefined fixed ROIs, the user can select his own ROIs at any moment. This information about the selected ROIs is sent to an extraction engine in order to customize the scalable bitstream and to generate an adapted bitstream (extraction process). Finally, the adapted bitstream can be decoded such that the ROI has a higher visual quality than the surrounding background (decoding process).

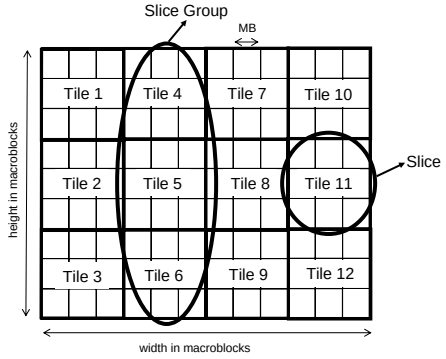

Figure 2. Frame divided in separately accessible tiles or slices

In H.264/AVC, the error resilience tool FMO can be used to define a certain ROI. FMO type 2 is the ideal candidate to support rectangular ROIs during the encoding phase [14]. Because the FMO maps containing the ROIs are defined during the encoding phase, this type cannot be used for IROI scalability without taking precautions.

In order to support interactive rectangular ROI scalability, a frame has to be divided into different tiles that can be selected individually. Such a division is shown in Figure 2, in which every tile represents a rectangle of 3 by 3 MacroBlocks (MBs). Each tile has to be coded as an individual slice such that the tiles can be decoded independently of other tiles. It is clear that the encoded bitstream contains no predefined ROIs and that the tiles belonging to a certain ROI can be selected on-the-fly during the extraction process.

There are several possibilities to generate bitstreams containing the tiled structure as shown in Figure 2. We will describe three possible configurations available in the current SVC specification for the encoder to obtain the desired partitioning in slices and tiles. The term Joint Scalable Video Model (JSVM) will be used further in this paper to indicate the current SVC specification.

- Using FMO type 2, it is possible to construct the tile structure of Figure 2. This type uses one or more rectangular slice groups and a background. For each vertical tile column (with a width of 3 macroblocks and a height of 9 macroblocks), a non-overlapping rectangular slice group is defined by coding the number of the top-left macroblock of the column (of the first row) and the bottom-right macroblock (of the last row). The last column is not defined by a rectangular slice group and represents the background of the frame.

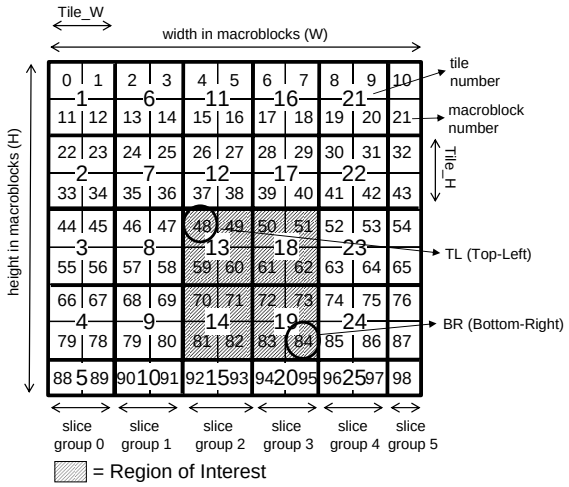

Figure 3. Frame divided in 6 slice groups

In order to obtain the horizontal division of the tile rows, a fixed number of macroblocks or rows for every slice in each slice group is given to the encoder as input parameter. For example, in Figure 2, nine macroblocks are used for each slice or tile.

- FMO type 0 (also called interleaved slice groups) is another slice group division that can be used to obtain our tiled slice partition. In case of this type, each slice group consists of a fixed number of macroblocks that it needs to contain in raster scan order before another slice group can be started. When all slice groups have been used and there are still some macroblocks left, the entire allocation process is repeated starting from the first slice group. To obtain our tiled partition, the width of the tile columns will be for the corresponding slice group. This way, the slice groups constitute a number of columns covering the frame.
  The height of the tiles can be obtained by giving the number of rows or a fixed number of macroblocks of a slice to the encoder. An example of such a division for a QCIF resolution video sequence is given in Figure 3.

- FMO type 6 can also be used to assign the macroblocks to different slice groups. In this case, a syntax element, `slice_group_id`, is sent for each macroblock. In other words, the complete *macroblock to slice group map* is encapsulated into the bitstream resulting in a lot of overhead compared to the other two methods.

2.2. MPEG-21 Bitstream Syntax Descriptions

MPEG-21 describes a multimedia framework that aims enabling a transparent and augmented use of multimedia resources across a wide range of networks and devices [5]. In such a framework, it should be possible to deliver scalable video content without the need to have knowledge regarding the underlying coding format. In order to meet the constraints of a particular usage environment, the bitstream customization process typically involves the removal of certain data chunks and the adaptation of the value of certain syntax elements.
In particular, part 7 of the MPEG-21 standard, the Digital Item Adaptation (DIA, [1]) specification, offers one way to realize this goal. This solution relies on automatically generated XML-based descriptions that contain information about the high-level bitstream structure. The DIA standard specifies MPEG-21 BSDL to describe the high-level structure of an encoded bitstream in XML. The generated documents are called Bitstream Syntax Descriptions (BSDs).
The entire chain of processes of the BSDL framework is given in Figure 4. Explanatory notes of the indicated numbers are provided below:

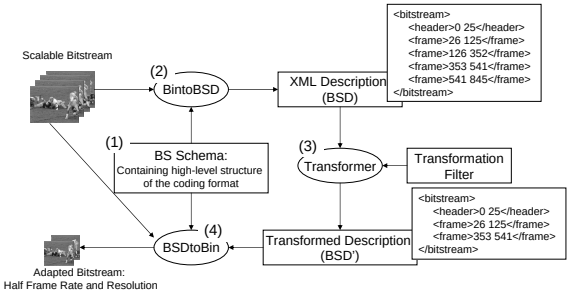

Figure 4. Functioning of the MPEG-21 BSDL framework

(1) The high-level structure of the coding format used is represented by a Bitstream Syntax Schema (BS Schema). The language used to construct such a BS Schema is BSDL, and this language is standardized in the DIA specification.

(2) The XML-based BSD of the scalable bitstream is generated by a format-agnostic parser once the original (encoded) bitstream and a corresponding BS Schema are known. The functioning of this parser, i.e. the BintoBSD Parser, is also defined the standard.

(3) The generated description (BSD) can subsequently be transformed by using a well-known XML transformation technology like Extensible Stylesheet Language

Transformations (XSLT, [13]) or Streaming Transformations for XML (STX, [6]). This transformation represents the adaptation process in the XML domain.

(4) The format-agnostic BSDtoBin Parser creates an adapted bitstream, using the transformed BSD, a corresponding BS Schema, and the original (scalable) bitstream.

Note that the employment of format-agnostic parsers (namely, BintoBSD and BSDtoBin) means that the code base of these parsers does not have to be rewritten in order to support the customization of bitstreams compliant with other coding formats.
In this paper, we translate the high-level structure of JSVM6-coded bitstreams into an XML document using BSDL. The following syntactical data structures are described in XML: Supplemental Enhancement Information (SEI) messages, Sequence Parameter Sets (SPSs), PPSs, Network Abstraction Layer (NAL) unit headers, and the first four syntax elements of the slices. This information is needed to exploit the IROI scalability in the XML domain, as further explained in Section 3. The development of the BS Schema used in order to generate the BSDs is discussed in [7].

3. Extraction of IROIs

3.1. Theoretical Extraction of a IROI

In order to enable the extraction of ROIs in an interactive manner, it is necessary that a bitstream is coded in such a way that the ROIs can be selected in an on-the-fly fashion. The scalability properties generated by a JSVM6-encoder [18] are shown in Figure 5. Bitstreams containing these properties are the subject of adaptation throughout this paper. As one can notice in the figure, every bitstream contains the three common scalability axes along which adaptations can be executed. In particular, the structure of each bitstream is as follows:

- two spatial layers to obtain spatial scalability;
- hierarchical B pictures to realize temporal scalable layers;
- and the spatial enhancement layer is composed of a quality base layer and three Fine Grain Scalability (FGS) enhancement layers to offer progressive refinement quality layers.

Bitstreams containing these characteristics are feasible candidates to be used in heterogeneous environments. In order to exploit IROI scalability, FMO type 0 is used in the spatial enhancement layer to obtain the tiled slice partition as explained in Section 2.1. This structure gives a user the possibility to select the tiles of the ROI on-the-fly, and the adaptation engine will only send the FGS enhancement layers belonging to the ROI to the decoder.

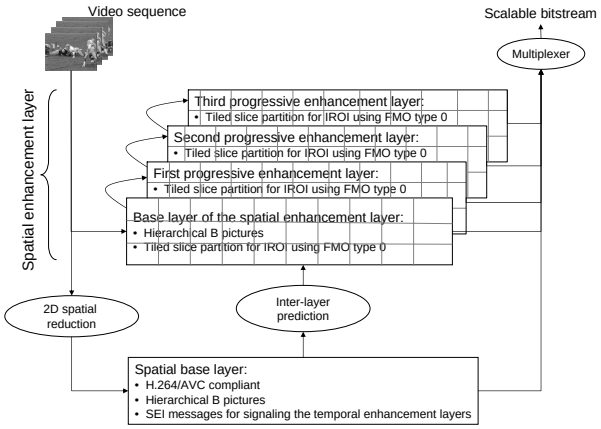

Figure 5. JSVM6 encoder generating IROI scalable bitstreams

Now, we will explain how the IROIs can be extracted from the original scalable bitstreams. Generally, the user will select the desired ROI by reporting a rectangular area. This area can preferably be described using the *FocusOfAttention* description tool, which is also a part of the MPEG-21 DIA standard [1]. The area is defined in terms of the top-left and bottom-right pixel coordinates. These coordinates are sent to the adaptation engine, which subsequently selects the slices that are part of the ROI. In the first step, the engine will map the pixel coordinates on the corresponding macroblock numbers, in particular by using following formula: $\lfloor y \text{ div } 16 \rfloor \times W + \lfloor x \text{ div } 16 \rfloor$. This number increases in raster scan order. The first macroblock of the frame has number 0 (see Figure 3). Once the macroblock numbers are calculated, the tile numbers, in which the top-left and bottom-right macroblocks of the ROI lie, can be determined. The tiles are numbered as indicated in Figure 2. These numbers can be calculated as follows (the abbreviations used throughout the formulas are shown in Figure 3):

$$
\begin{aligned}
TL &= \text{number of top-left macroblock of ROI} \\
BR &= \text{number of bottom-right macroblock of ROI} \\
W &= \text{width of the frame in terms of macroblocks} \\
H &= \text{height of the frame in terms of macroblocks} \\
Tile\_W &= \text{width of a tile in terms of macroblocks} \\
Tile\_H &= \text{height of a tile in terms of macroblocks}
\end{aligned}
$$

$$
\begin{aligned}
Tile\_id\_TL &= (\lfloor (TL \bmod W) \text{ div } Tile\_W \rfloor \times \lceil H \text{ div } Tile\_H \rceil) \\
&\quad + (\lfloor \lfloor TL \text{ div } W \rfloor \text{ div } Tile\_H \rfloor + 1) \\
Tile\_id\_BR &= (\lfloor (BR \bmod W) \text{ div } Tile\_W \rfloor \times \lceil H \text{ div } Tile\_H \rceil) \\
&\quad + (\lfloor \lfloor BR \text{ div } W \rfloor \text{ div } Tile\_H \rfloor + 1)
\end{aligned}
$$

Once the top-left and bottom-right corner of the ROI are calculated, then all other tile numbers representing the area of the desired ROI can be determined. A tile with tile number $Tile\_i, 1 \leq Tile\_i \leq \lceil W \text{ div } Tile\_W \rceil \times \lceil H \text{ div } Tile\_H \rceil$ is a part of the ROI if:

$$\begin{gathered}((Tile\_i-1) \bmod H\_Tiles \geq (Tile\_id\_TL-1) \bmod H\_Tiles) \\ \wedge((Tile\_i-1) \bmod H\_Tiles \leq (Tile\_id\_BR-1) \bmod H\_Tiles) \\ \wedge(\lceil Tile\_i \text{ div } H\_Tiles \rceil \geq \lceil Tile\_id\_TL \text{ div } H\_Tiles \rceil) \\ \wedge(\lceil Tile\_i \text{ div } H\_Tiles \rceil \leq \lceil Tile\_id\_BR \text{ div } H\_Tiles \rceil) \\ \text{in which: } H\_Tiles = \lceil H \text{ div } Tile\_H \rceil\end{gathered}$$

The frame division in separately accessible tiles is an abstraction layer on top of the macroblock division. Information about this abstraction layer is in a limited way present in the scalable bitstream. In particular, there is only information available about the width and height of the tiles. In order to determine the location of a slice in a frame, the syntax element `first_mb_in_slice` can be used. The last step for the adaptation engine consists of the mapping of each tile that is part of the ROI on the value of the syntax element `first_mb_in_slice` of the corresponding slice. The value of syntax element `first_mb_in_slice` for a tile with number $Tile\_i$ can be obtained as follows:

$$\begin{aligned}&W \times ((Tile\_i-1) \bmod \lceil H \text{ div } Tile\_H \rceil) \times Tile\_H \\ &+Tile\_W \times (\lfloor (Tile\_i-1) \text{ div } \lceil H \text{ div } Tile\_H \rceil \rfloor)\end{aligned}$$

Now, all necessary information is available in the adaptation engine in order to select the slices that are part of the ROI by comparing the value of `first_mb_in_slice` of a slice with the values of the ROI.

3.2. High-level structure of an SVC bitstream

The structure of a scalable bitstream is given in Figure 6 and this structure can be found again in the XML descriptions (BSDs). This high-level structure will now be discussed to be able to explain the BSD transformations.

A scalable bitstream can contain SEI messages. Such a message assists in the processes related to decoding, display or other purposes and can be ignored by a decoder to reproduce the luma and chroma samples. SEI messages can for example be used for signaling information related to buffering, timing, cropping, and so on.

The first SEI message in the scalable bitstreams used (as shown in Figure 6) is the `scalability_info` SEI message. This message contains information about the scalability axes incorporated in the bitstream such as the number of spatial layers, the resolution of the layers, the number of temporal levels, frame rates, information about the grids for IROI, etc. Based on this message, an adaptation engine can build up the internal structure of the bitstream. The engine uses this information to decide which kind of tailored bitstreams can be generated from the parent stream.

After this first SEI message, a number of SPSs follow; namely at least one SPS for each spatial layer. An SPS is applicable to a complete sequence of pictures of a particular spatial layer and contains information about the profile, the spatial resolution of the pictures in the sequence, etc.

Furthermore, a number of PPSs are encapsulated in the bitstream. A PPS is applicable to a number of pictures of a sequence. There exist at least as many PPSs as SPSs in the bitstream because every SPS must be referred by a PPS. The PPS belonging to the spatial enhancement layer contains, among other things, the number of slice groups, the type of the slice group map, and the value of the `run_length_minus1` syntax element in order to obtain the tiled slice division. Normally, after having received the needed SPSs and PPSs, the decoder can start reproducing the video sequence by decoding the coded pixel values. In our scalable bitstreams other SEI messages are present as well, in particular `sub_seq_info` SEI messages. These SEI messages contain information about the temporal decomposition of the video sequence. Finally, the NAL units containing a slice of a certain frame will follow. In particular, first, the slices of the spatial base layer are embedded in the bitstream, followed by the slices of the tiled division of the spatial enhancement layer and the progressive refinement layers. A subsequent `sub_seq_info` SEI message indicates the start of the next frame.

3.3. Extraction in the XML domain

The structure of the SVC coding format, as discussed in Section 3.2 for the bitstreams used in this research, will be established in a BS Schema using BSDL. The high-level structure of a coded bitstream can be found in the BSD after executing the BintoBSD process, which is guided by the BS Schema. The two most important data blocks in the bitstream structure for an IROI-based adaptation engine are the `scalability_info` SEI message and the value of the `first_mb_in_slice` syntax element (available in the slice header).

Figure 7 contains a part of the `scalability_info` SEI message as available in the used BSDs. This message is composed of information about the different layers that can be extracted. The number of layers is indicated on line 3 (in particular, 25 layers in this BSD[1]). In Figure 7, information about a possible layer is shown. From line 9 up to and including line 11, information about the temporal, spatial, and quality level of a layer is given. This layer belongs to the quality base layer of the second spatial level, having the lowest temporal decomposition.

Further, information about the bit rate, frame rate, and res-

[1] 25 because of 5 temporal layers for each spatial and quality layer

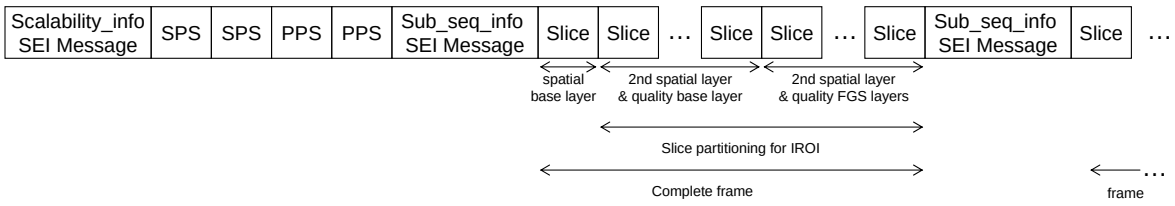

Figure 6. Structure of the bitstream generated by the encoder of Figure 5

olution of the layer is given. The most important part for our IROI-based adaptation engine is the information about the tiled slice division. This information is shown in the figure from line 42 until 48 inclusive. It gives the height and width of the tiles in terms of macroblocks (in our example, grids of 6x6 macroblocks). Based on this information, the adaptation engine calculates the values of the `first_mb_in_slice` syntax element of the ROI that has to be extracted as discussed in Section 3.1.
Once the `scalability_info` SEI message is interpreted and the tiled slice division is calculated, the slices belonging to the IROI can be selected. This requires an examination of the first syntax elements of the slice headers. In Figure 8, the XML representation of two slices is given.

- The first slice (line 4 until 23) belongs to the H.264/AVC-compliant spatial base layer. This classification of the slice is based on the NAL unit type (type 1 on line 7). As one can see, this slice contains no information about the temporal level to which it belongs. Therefore, a `sub_seq_info` SEI message will precede this NAL unit (which is not shown in the figure and which is less important for the IROI extraction). More information about this SEI message can be found in [16]. In case the adaptation engine detects such a slice, the description of this slice will be copied to the transformed BSD (the complete spatial base layer will be present in the transformed BSD and so also in the adapted bitstream).

- The second part of the fragment in Figure 8 shows a slice that belongs to a scalable enhancement layer (indicated by NAL unit type 20 on line 27). Such a NAL unit also contains information about the scalability properties of the layer to which the slice belongs. This information is present in the NAL unit header from line 28 to 35. The value of the syntax element `first_mb_in_slice` is also shown (line 40). The transformation engine uses this value to identify whether the slice belongs to the desired ROI or not.

```
1   <?xml version="1.0"?>
    <scalability_info>
      <num_layers_minus1>24</num_layers_minus1>
      <!-- Previous layers -->
5     <layer_info>
        <layer_id>5</layer_id>
        <simple_priority_id>0</simple_priority_id>
        <discardable_flag>0</discardable_flag>
        <temporal_level>0</temporal_level>
10      <dependency_id>1</dependency_id>
        <quality_level>0</quality_level>
        <sub_pic_layer_flag>0</sub_pic_layer_flag>
        <sub_region_layer_flag>0</sub_region_layer_flag>
        <iroi_slice_division_info_present_flag>1</
              iroi_slice_division_info_present_flag>
15      <profile_level_info_present_flag>0</profile_level_info_present_flag>
        <bitrate_info_present_flag>1</bitrate_info_present_flag>
        <frm_rate_info_present_flag>1</frm_rate_info_present_flag>
        <frm_size_info_present_flag>1</frm_size_info_present_flag>
        <layer_dependency_info_present_flag>1</
              layer_dependency_info_present_flag>
20      <init_parameter_sets_info_present_flag>0</
              init_parameter_sets_info_present_flag>
        <exact_interlayer_pred_flag>1</exact_interlayer_pred_flag>
        <profile_level_info_present_flag_is_0>
          <profile_level_info_src_layer_id_delta>0</
                profile_level_info_src_layer_id_delta>
        </profile_level_info_present_flag_is_0>
25      <bitrate_info_present_flag_is_1>
          <avg_bitrate>11</avg_bitrate>
          <max_bitrate_layer>0</max_bitrate_layer>
          <max_bitrate_decoded_picture>0</max_bitrate_decoded_picture>
          <max_bitrate_calc_window>0</max_bitrate_calc_window>
30      </bitrate_info_present_flag_is_1>
        <frm_rate_info_present_flag_is_1>
          <constant_frm_rate_idc>0</constant_frm_rate_idc>
          <avg_frm_rate>480</avg_frm_rate>
        </frm_rate_info_present_flag_is_1>
35      <frm_size_info_present_flag_is_1>
          <frm_width_in_mbs_minus1>21</frm_width_in_mbs_minus1>
          <frm_height_in_mbs_minus1>17</frm_height_in_mbs_minus1>
        </frm_size_info_present_flag_is_1>
        <sub_region_layer_flag_is_0>
40        <sub_region_info_src_layer_id_delta>0</
                sub_region_info_src_layer_id_delta>
        </sub_region_layer_flag_is_0>
        <iroi_slice_division_info_present_flag_eq_1>
          <iroi_slice_division_type>0</iroi_slice_division_type>
          <if_iroi_slice_division_type_is_0>
45          <grid_slice_width_in_mbs_minus1>5</
                  grid_slice_width_in_mbs_minus1>
            <grid_slice_height_in_mbs_minus1>5</
                  grid_slice_height_in_mbs_minus1>
          </if_iroi_slice_division_type_is_0>
        </iroi_slice_division_info_present_flag_eq_1>
        <layer_dependency_info_present_flag_is_1>
50        <num_directly_dependent_layers>1</num_directly_dependent_layers>
          <directly_dependent_layer_id_delta_i_j>5</
                directly_dependent_layer_id_delta_i_j>
        </layer_dependency_info_present_flag_is_1>
        <init_parameter_sets_info_present_flag_is_0>
          <init_parameter_sets_info_src_layer_id_delta>0</
                init_parameter_sets_info_src_layer_id_delta>
55      </init_parameter_sets_info_present_flag_is_0>
      </layer_info>
      <!-- Following layers -->
    </scalability_info>
```

Figure 7. Fragment of a `scalability_info` SEI message as available in the BSDs

Note that the slice is not completely described in the XML document; only the first syntax elements are available, needed for guiding the decision-making process. The other syntax elements and the coded pixel values are encapsulated in the slice payload (line 43). This line represents a byte range indicated by the start byte and the length of the pay-

load. These values point to the original bitstream (this is the reason why the original bitstream is needed by the BSDtoBin Parser in Figure 4, in particular to copy the payloads from the original bitstream to the adapted one).

```
<?xml version="1.0"?>
<bitstream>
  <!-- NAL unit of a slice belonging to the spatial base layer -->
  <nal_unit>
    <forbidden_zero_bit>0</forbidden_zero_bit>
    <nal_ref_idc>3</nal_ref_idc>
    <nal_unit_type>1</nal_unit_type>
    <raw_byte_sequence_payload>
      <coded_slice_of_a_non_IDR_picture>
        <slice_layer_without_partitioning_rbsp>
          <slice>
            <first_mb_in_slice>0</first_mb_in_slice>
            <slice_type>0</slice_type>
            <pic_parameter_set_id>0</pic_parameter_set_id>
            <frame_num xsi:type="b9">1</frame_num>
            <bit_stuffing>1</bit_stuffing>
            <slice_payload>18016 4291</slice_payload>
          </slice>
        </slice_layer_without_partitioning_rbsp>
      </coded_slice_of_a_non_IDR_picture>
    </raw_byte_sequence_payload>
  </nal_unit>
  <!-- NAL unit of a slice belonging to the spatial enhancement layer-->
  <nal_unit>
    <forbidden_zero_bit>0</forbidden_zero_bit>
    <nal_ref_idc>3</nal_ref_idc>
    <nal_unit_type>20</nal_unit_type>
    <nal_unit_information_for_scalable_extension>
      <simple_priority_id>0</simple_priority_id>
      <discardable_flag>0</discardable_flag>
      <reserved_zero_bit>0</reserved_zero_bit>
      <temporal_level>0</temporal_level>
      <dependency_id>1</dependency_id>
      <quality_level>0</quality_level>
    </nal_unit_information_for_scalable_extension>
    <raw_byte_sequence_payload>
      <coded_slice_of_a_non_IDR_picture_in_scalable_extension>
        <slice_layer_in_scalable_extension_rbsp>
          <slice_in_scalable_extension>
            <first_mb_in_slice>132</first_mb_in_slice>
            <slice_type>1</slice_type>
            <bit_stuffing>24</bit_stuffing>
            <slice_payload>22387 41</slice_payload>
          </slice_in_scalable_extension>
        </slice_layer_in_scalable_extension_rbsp>
      </coded_slice_of_a_non_IDR_picture_in_scalable_extension>
    </raw_byte_sequence_payload>
  </nal_unit>
</bitstream>
```

Figure 8. Fragments of slice representations

4. Experimental Results

4.1. Methodology

To evaluate the performance of our XML-driven adaptation framework for the exploitation of IROI scalability, we have generated two scalable bitstreams. These bitstreams contain the scalability axes as shown in Figure 5. The two sequences are the well-known Crew and Ice sequences, having a resolution of 704×576 at the highest (second) spatial layer and containing 150 frames at full frame rate (30Hz). In each scalable bitstream, the spatial base layer is coded with a quantization parameter of 25. The quality base layer of the spatial enhancement layer is coded with a quantization parameter of 45. This quality base layer is coded with FMO type 0 and is extended with three FGS refinement layers. FMO is used to obtain the tiled slice division. Each tile has a size of 7 by 6 macroblocks (except the tiles belonging to last tile column or row) resulting in 7 slice groups (of which the first 6 slice groups have 6 as value for the `run_length_minus1` syntax element and 1 for the last slice group). In the experiments, we have used version 5.12 of the JSVM6 reference software.

For each scalable bitstream, the corresponding BSD has to be generated in order to extract the desired ROIs. Therefore, we have used an optimized BintoBSD Parser the functioning of which is discussed in [8]. The generated BSDs are the subject of a transformation performed by an adaptation engine. This engine typically takes a rectangular ROI from the user as input to know the most interesting part of the video sequence. In our experiments, the ROI is defined by pixel coordinates $(368; 64)$ and $(560; 272)$. The transformation engine selects the complete spatial base layer and the quality base layer of the spatial enhancement layer (in order to keep the background on which the motion estimation was executed), together with the three FGS layers for the slices belonging to the ROI. The BSD transformation engine will be steered by a stylesheet that implements our algorithm as discussed in Section 3. There exist multiple technologies to implement an XML transformation and we have chosen to use the STX language. This choice was inspired by the streaming capabilities and low memory footprint of STX [9]. Finally, the adapted bitstreams are created from the transformed BSDs by using the BSDtoBin Parsers as implemented in the MPEG-21 DIA reference software [3].

The measurements were done on a PC having an Intel Pentium D CPU, clocked at 2.8GHz and having 2GB of RAM at its disposal. The operating system used was Windows XP Pro (service pack 2). Sun Microsystems Java 2 Runtime Environment (Standard Edition version 1.5.0 02-b09) was running as Java Virtual Machine (needed to execute the BSDL and BSD transformation software). The STX engine used in our tests is *Joost* (version 2005-05-21)[2]. Finally, all time measurements were executed five times. An average was calculated over the five runs.

4.2. Discussion and Results

The results of our experiments are shown in Table 1. The first part of the table contains information about the generated bitstreams. Each sequence was encoded two times, once with and once without FMO. One can observe that the overhead of using FMO is significant while the same quality is kept, in particular 31.7% for the Crew sequence and 42.7% for the Ice sequence. This overhead can be attributed to a less efficient intra prediction (which needs to be done inside the small slices), as well as to the enormous number of extra NAL unit and slice headers that have to be encapsulated in the bitstream. More precisely, for each frame, 4 headers are needed in the spatial enhancement layer in case

[2]This engine can be found on `http://joost.sourceforge.net`.

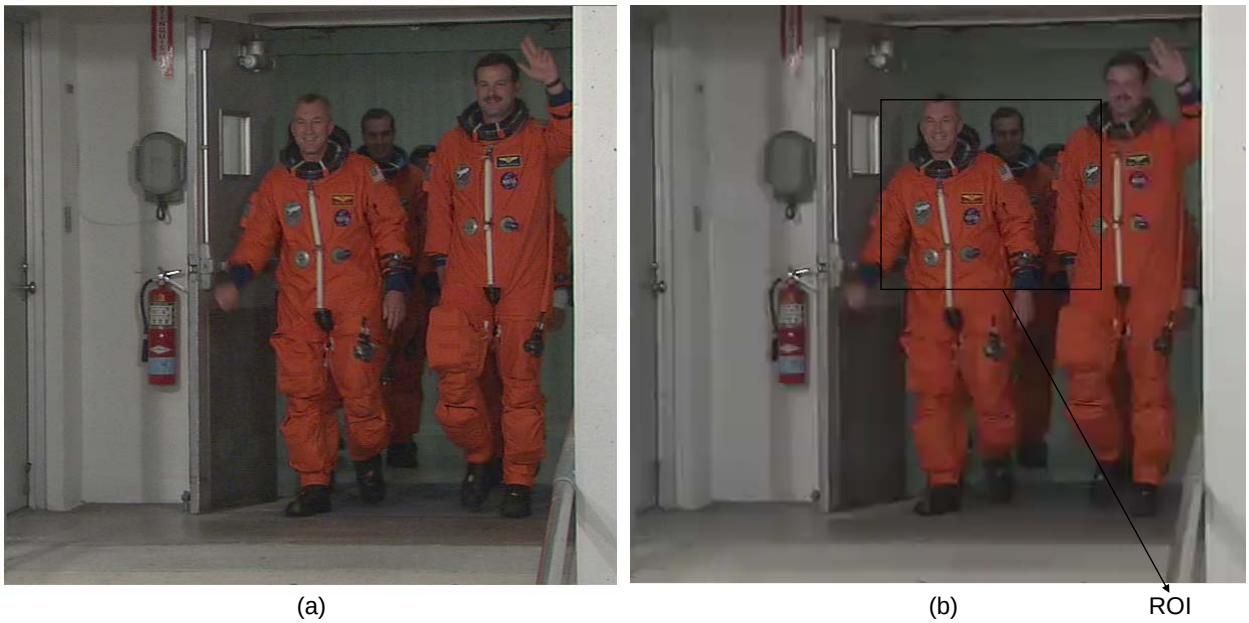

Figure 9. Example of the ROI selection: (a) is the original coded bitstream, (b) is the adapted version

no FMO is used, while 168 headers are necessary to obtain the desired tiled slice division when FMO is in use. Furthermore, this overhead is also content dependent. The more complex the sequence, the lesser the impact of the header on the bitstream size (the Crew sequence is more complex than the Ice sequence). Note that only FMO-encoded bitstreams are suited for IROI extraction.

The second part of the table contains the performance of our XML-driven adaptation framework. First, one can notice that the BSD generation process takes the most time. Nevertheless, the execution time is acceptable in comparison with the encoding time (multiple hours) and this process has to be executed only once. The generated BSDs are verbose. As one can see, they are multiple times larger than the corresponding bitstream itself. A solution for this problem is to compress the XML documents, for example, by using Binary MPEG format for XML (BiM, [4]). In the table, we have given the results by compressing the documents using the default text compression algorithm of WinRAR (it is expected that these results give a good indication for tools built on top of BiM, [17]). However, this process cannot be executed in real time. The origin of this behavior is again the verbose nature of the BSDs. The STX engine spends too much time on I/O operations. More precisely, the STX engine parses approximately 1600 NAL units per second. The sizes of the transformed BSDs and their compressed equivalents are three times smaller than the original ones. This is as expected because of the removal of the FGS layers from the slices that do not belong to the ROI. Finally, the generation of the adapted bitstream from the transformed BSD is the fastest process: it can be done in almost real time. This process is faster than the transformation because less NAL units have to be interpreted.

The last part of the table shows the bit rate and quality of the adapted bitstreams after the extraction of the ROIs. One can see that the bit rate reduces significantly without a dramatical reduction in objective quality. Note that the quality of the ROI is still intact; only the background has a degraded quality. This is shown in Figure 9. The left frame contains the original coded bitstream. The right frame of the figure contains the adapted version in which the ROI is selected and is transmitted at full quality while the background has a degraded quality.

5. Conclusions and future research

In this paper, we have shown how Interactive Region of Interest scalability can be obtained in the Scalable extension of H.264/AVC. IROI means that the ROI is not coded during the encoding process but that this area of interest can be extracted during an adaptation process. As such, the IROI can be selected on-the-fly. Therefore, the encoded scalable bitstreams should be structured such that these ROIs can be selected and extracted without a complete decode-encode step. In this paper, we explain how such a structure can be obtained. An abstraction layer on top of the frames is defined which divides the frames in a tiled structure such that each tile represents an individual accessible slice. The FMO

Table 1. Performance results

Process	Crew sequence	Ice sequence
Bit rate without FMO (KB)	1508	1055
Quality without FMO (dB)	38.31	41.15
Bit rate with FMO (KB)	1987	1506
Quality with FMO (dB)	38.3	41.17
BSD generation (s)	75.2	78.02
BSD size (KB)	29340	29335
Compressed BSD size (KB)	581	579
Transformation (s)	16.7	16.7
Transformed BSD size (KB)	10010	10008
Compressed transformed BSD size (KB)	220	219
Bitstream generation (s)	6.41	6.34
Bit rate extracted (KB)	852	681
Quality adapted bitstream (dB)	34.43	35.77

tool was used to create this tiled structure. After a discussion of the generation of IROI scalable bitstreams, we outline the extraction process such that the ROIs are extracted at a high visual quality, while the surrounding background has a degraded quality. This extraction process can be executed on the scalable bitstream itself. However, in this paper, the adaptation step is executed in the XML domain. The XML transformation is implemented in STX and gives us the opportunity to obtain a format-agnostic video content adaptation framework in which ROIs can be selected in an on-the-fly fashion.

From the experimental results, we can conclude that the overhead of using FMO is significant, as well as the sizes of the XML descriptions on which transformations are to be executed. These shortcomings of our adaptation framework will be examined in future research. Furthermore, we will investigate the possibility to change the IROI during the BSD transformation and streaming process.

6. 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] ISO/IEC 21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation, 2004.

[2] ISO/IEC 14496-10:2005 Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding, 2005.

[3] ISO/IEC 21000-8:2006 Information technology – Multimedia framework (MPEG-21) – Part 8: Reference Software, 2006.

[4] ISO/IEC 23001-1:2006 Information technology – MPEG systems technologies – Part 1: Binary MPEG format for XML, 2006.

[5] I. S. Burnett, F. Pereira, R. Van de Walle, and R. Koenen. *The MPEG-21 Book*. Wiley, John & Sons, Inc, 2006.

[6] P. Cimprich et al. Streaming transformations for XML version 1.0 working draft. `http://stx.sourceforge.net/documents/spec-stx-20040701.html`.

[7] D. De Schrijver, W. De Neve, K. De Wolf, S. Notebaert, and R. Van de Walle. XML-based customization along the scalability axes of H.264/AVC scalable video coding. In *Proceedings of 2006 IEEE International Symposium on Circuits and Systems*, pages 465–468, Island of Kos, Greece, May 2006.

[8] D. De Schrijver, W. De Neve, K. De Wolf, and R. Van de Walle. Generating MPEG-21 BSDL descriptions using context-related attributes. In *Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 79–86, Irvine, CA, December 2005.

[9] 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. In *Proceedings of 2006 IEEE International Conference on Innovative Computing, Information and Control*, volume 3, pages 213–216, Beijing, China, August 2006.

[10] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner. Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, 7(3):463–470, June 2005.

[11] A. Ebrahimi-Moghadam and S. Shirani. Progressive scalable interactive region-of-interest image coding using vector quantization. *IEEE Transactions on Multimedia*.

[12] ISO/IEC JTC/SC29/WG11. Applications and Requirements for Scalable Video Coding. *N6880*, January 2005.

[13] M. Kay. *XSLT Programmer's Reference, 2nd edition*. Wrox Press Ltd., Birmingham, UK, 2001.

[14] P. Lambert, W. De Neve, D. De Schrijver, Y. Dhondt, and R. Van de Walle. Using Placeholder Slices and MPEG-21 BSDL for ROI Extraction in H.264/AVC FMO-encoded Bitstreams. In *Proceedings of SIGMAP 2006*, pages 9–16, Setúbal, Portugal, August 2006.

[15] 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*, 18(8):721–747, September 2003.

[16] D. Tian, M. M. Hannuksela, and M. Gabbouj. Sub-sequence video coding for improved temporal scalability. In *IEEE International Symposium on Circuits and Systems, 2005*, volume 6, pages 6074–6077, Kobe, Japan, 5 2005.

[17] C. Timmerer, I. Kofler, J. Liegl, and H. Hellwagner. An evaluation of existing metadata compression and encoding technologies for MPEG-21 applications. In *Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 534–539, Irvine, CA, December 2005.

[18] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien. Scalable Video Coding - Joint Draft 6. *Doc. JVT-S201*, April 2006.