

## INTER-LAYER REFERENCE PICTURE PROCESSING FOR CODING- STANDARD SCALABILITY

### CROSS-REFERENCE TO RELATED APPLICATIONS

- 5 [0001] This application claims priority to United States Provisional Patent Application No. 61/706,480 filed 27 September 2012, which is hereby incorporated by reference in its entirety.

### TECHNOLOGY

- 10 [0002] The present invention relates generally to images. More particularly, an embodiment of the present invention relates to inter-layer reference picture processing for coding-standard scalability.

### BACKGROUND

- 15 [0003] Audio and video compression is a key component in the development, storage, distribution, and consumption of multimedia content. The choice of a compression method involves tradeoffs among coding efficiency, coding complexity, and delay. As the ratio of processing power over computing cost increases, it allows for the development of more complex compression techniques that allow for more efficient compression. As an example, in video compression, the Motion Pictures Expert Group (MPEG) from the International Standards Organization (ISO) has  
20 continued improving upon the original MPEG-1 video standard by releasing the MPEG-2, MPEG-4 (part 2), and H.264/AVC (or MPEG-4, part 10) coding standards.

- [0004] Despite the compression efficiency and success of H.264, a new generation of video compression technology, known as High Efficiency Video Coding (HEVC), is now under development. HEVC, for which a draft is available in  
25 “*High efficiency video coding (HEVC) text specification draft 8*,” ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-J1003, July 2012, by B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, which is incorporated herein by reference in its entirety, is expected to provide improved compression capability over the existing H.264 (also known as AVC) standard,  
30 published as, “*Advanced Video Coding for generic audio-visual services*,” ITU T Rec. H.264 and ISO/IEC 14496-10, which is incorporated herein in its entirety. As appreciated by the inventors here, it is expected that over the next few years H.264 will still be the dominant video coding standard used worldwide for the distribution of

digital video. It is further appreciated that newer standards, such as HEVC, should allow for backward compatibility with existing standards.

[0005] As used herein, the term “coding standard” denotes compression (coding) and decompression (decoding) algorithms that may be both standard-based, open-  
5 source, or proprietary, such as the MPEG standards, Windows Media Video (WMV), flash video, VP8, and the like.

[0006] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of  
10 the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section. Similarly, issues identified with respect to one or more approaches should not assume to have been recognized in any prior art on the basis of this section, unless otherwise indicated.

#### BRIEF DESCRIPTION OF THE DRAWINGS

15 [0007] An embodiment of the present invention is illustrated by way of example, and not in way by limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0008] FIG. 1 depicts an example implementation of a coding system supporting coding-standard scalability according to an embodiment of this invention;

20 [0009] FIG. 2A and FIG. 2B depict example implementations of a coding system supporting AVC/H.264 and HEVC codec scalability according to an embodiment of this invention;

[0010] FIG. 3 depicts an example of layered coding with a cropping window according to an embodiment of this invention;

25 [0011] FIG. 4 depicts an example of inter-layer processing for interlaced pictures according to an embodiment of this invention;

[0012] FIG. 5A and FIG. 5B depict examples of inter-layer processing supporting coding-standard scalability according to an embodiment of this invention;

[0013] FIG. 6 depicts an example of RPU processing for signal encoding model  
30 scalability according to an embodiment of this invention;

[0014] FIG. 7 depicts an example encoding process according to an embodiment of this invention;

[0015] FIG. 8 depicts an example decoding process according to an embodiment of this invention; and

[0016] FIG. 9 depicts an example decoding RPU process according to an embodiment of this invention.

#### DESCRIPTION OF EXAMPLE EMBODIMENTS

[0017] Inter-layer reference picture processing for coding-standard scalability is described herein. Given a base layer signal, which is coded by a base layer (BL) encoder compliant to a first coding standard (e.g., H.264), a reference processing unit (RPU) process generates reference pictures and RPU parameters according to the characteristics of input signals in the base layer and one or more enhancement layers. These inter-layer reference frames may be used by an enhancement layer (EL) encoder which is compliant to a second coding standard (e.g., HEVC), to compress (encode) one or more enhancement layer signals, and combine them with the base layer to form a scalable bit stream. In a receiver, after decoding a BL stream with a BL decoder which is compliant to the first coding standard, a decoder RPU may apply received RPU parameters to generate inter-layer reference frames from the decoded BL stream. These reference frames may be used by an EL decoder which is compliant to the second coding standard to decode the coded EL stream.

[0018] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are not described in exhaustive detail, in order to avoid unnecessarily obscuring the present invention.

#### OVERVIEW

[0019] Example embodiments described herein relate to inter-layer reference picture processing for coding-standard scalability. In one embodiment, video data are coded in a coding-standard layered bit stream. Given base layer (BL) and enhancement layer (EL) signals, the BL signal is coded into a BL stream using a BL encoder which is compliant to a first encoding standard. In response to the BL signal and the EL signal, a reference processing unit (RPU) determines RPU processing parameters. In response to the RPU processing parameters and the BL signal, the RPU generates an inter-layer reference signal. Using an EL encoder which is compliant to a second coding standard, the EL signal is coded into a coded EL stream, where the encoding of the EL signal is based at least in part on the inter-layer reference signal.

[0020] In another embodiment, a receiver demultiplexes a received scalable bitstream to generate a coded BL stream, a coded EL stream, and an RPU data stream. A BL decoder compliant to a first coding standard decodes the coded BL stream to generate a decoded BL signal. A receiver with an RPU may also decode the RPU data stream to determine RPU process parameters. In response to the RPU processing parameters and the BL signal, the RPU may generate an inter-layer reference signal. An EL decoder compliant to a second coding standard may decode the coded EL stream to generate a decoded EL signal, where the decoding of the coded EL stream is based at least in part on the inter-layer reference signal.

10 LAYERED-BASED CODING-STANDARD SCALABILITY

[0021] Compression standards such as MPEG-2, MPEG-4 (part 2), H.264, flash, and the like are being used word-wide for delivering digital content through a variety of media, such as, DVD discs or Blu-ray discs, or for broadcasting over the air, cable, or broadband. As new video coding standards, such as HEVC, are developed, adoption of the new standards could be increased if they would support some backward compatibility with existing standards.

[0022] FIG. 1 depicts an embodiment of an example implementation of a system supporting coding-standard scalability. The encoder comprises a base layer (BL) encoder (110) and an enhancement layer (EL) encoder (120). In an embodiment, BL Encoder 110 is a legacy encoder, such as an MPEG-2 or H.264 encoder, and EL Encoder 120 is a new standard encoder, such as an HEVC encoder. However, this system is applicable to any combination of either known or future encoders, whether they are standard-based or proprietary. The system can also be extended to support more than two coding standards or algorithms.

[0023] According to FIG. 1, an input signal may comprise two or more signals, e.g., a base layer (BL) signal 102 and one or more enhancement layer (EL) signals, e.g. EL 104. Signal BL 102 is compressed (or coded) with BL Encoder 110 to generate a coded BL stream 112. Signal EL 104 is compressed by EL encoder 120 to generate coded EL stream 122. The two streams are multiplexed (e.g., by MUX 125) to generate a coded scalable bit stream 127. In a receiver, a demultiplexor (DEMUX 130) may separate the two coded bit streams. A legacy decoder (e.g., BL Decoder 140) may decode only the base layer 132 to generate a BL output signal 142. However, a decoder that supports the new encoding method (EL Encoder 120), may also decode the additional information provided by the coded EL stream 134 to

generate EL output signal 144. BL decoder 140 (e.g., an MPEG-2 or H.264 decoder) corresponds to the BL encoder 110. EL decoder 150 (e.g., an HEVC decoder) corresponds to the EL Encoder 120.

[0024] Such a scalable system can improve coding efficiency compared to a simulcast system by properly exploring inter-layer prediction, that is, by coding the enhancement layer signal (e.g., 104) by taking into consideration information available from the lower layers (e.g., 102). Since the BL Encoder and EL Encoder comply to different coding standards, in an embodiment, coding standard-scalability may be achieved through a separate processing unit, the encoding reference processing unit (RPU) 115.

[0025] RPU 115 may be considered an extension of the RPU design described in PCT Application PCT/US2010/040545, "*Encoding and decoding architecture for format compatible 3D video delivery*," by A. Tourapis, et al., filed on June 30, 2010, and published as WO 2011/005624, which is incorporated herein by reference for all purposes. The following descriptions of the RPU apply, unless otherwise specified to the contrary, both to the RPU of an encoder and to the RPU of a decoder. Artisans of ordinary skill in fields that relate to video coding will understand the differences, and will be capable of distinguishing between encoder-specific, decoder-specific and generic RPU descriptions, functions and processes upon reading of the present disclosure. Within the context of a video coding system as depicted in FIG. 1, the RPU (115) generates inter-layer reference frames based on decoded images from BL Encoder 110, according to a set of rules of selecting different RPU filters and processes.

[0026] The RPU 115 enables the processing to be adaptive at a region level, where each region of the picture/sequence is processed according to the characteristics of that region. RPU 115 can use horizontal, vertical, or two dimensional (2D) filters, edge adaptive or frequency based region-dependent filters, and/or pixel replication filters or other methods or means for interlacing, de-interlacing, filtering, up-sampling, and other image processing.

[0027] An encoder may select RPU processes and outputs regional processing signals, which are provided as input data to a decoder RPU (e.g., 135). The signaling (e.g., 117) may specifies the processing method on a per-region basis. For example, parameters that relate to region attributes such as the number, size, shape and other characteristics may be specified in an RPU-data related data header. Some of the

filters may comprise fixed filter coefficients, in which case the filter coefficients need not be explicitly signaled by the RPU. Other processing modes may comprise explicit modes, in which the processing parameters, such as coefficient values are signaled explicitly. The RPU processes may also be specified per each color component.

5   **[0028]**     The RPU data signaling 117 can either be embedded in the encoded bitstream (e.g., 127), or transmitted separately to the decoder. The RPU data may be signaled along with the layer on which the RPU processing is performed.

10   Additionally or alternatively, the RPU data of all layers may be signaled within one RPU data packet, which is embedded in the bit stream either prior to or subsequent to embedding EL encoded data. The provision of RPU data may be optional for a given layer. In the event that RPU data is not available, a default scheme may thus be used for up-conversion of that layer. Not dissimilarly, the provision of an enhancement layer encoded bit stream is also optional.

15   **[0029]**     An embodiment allows for multiple possible methods of selecting processing steps within an RPU. A number of criteria may be used separately or in conjunction in determining RPU processing. The RPU selection criteria may include the decoded quality of the base layer bitstream, the decoded quality of the enhancement layer bitstreams, the bit rate required for the encoding of each layer including the RPU data, and/or the complexity of decoding and RPU processing of the data.

20   **[0030]**     The RPU 115 may serve as a pre-processing stage that processes information from BL encoder 110, before utilizing this information as a potential predictor for the enhancement layer in EL encoder 120. Information related to the RPU processing may be communicated (e.g., as metadata) to a decoder as depicted in FIG. 1 using an RPU Layer stream 136. RPU processing may comprise a variety of image processing operations, such as: color space transformations, non-linear quantization, luma and chroma up-sampling, and filtering. In a typical implementation, the EL 122, BL 112, and RPU data 117 signals are multiplexed into a single coded bitstream (127).

25   **[0031]**     Decoder RPU 135 corresponds to the encoder RPU 115, and with guidance from RPU data input 136, may assist in the decoding of the EL layer 134 by performing operations corresponding to operations performed by the encoder RPU 115.

[0032] The embodiment depicted in FIG. 1 can easily be extended to support more than two layers. Furthermore, it may be extended to support additional scalability features, including: temporal, spatial, SNR, chroma, bit-depth, and multi-view scalability.

5

#### H.264 and HEVC Coding-standard Scalability

[0033] In an example embodiment, FIG. 2A and FIG. 2B depict an example embodiment for layer-based coding-standard scalability as it may be applied to the HEVC and H.264 standards. Without loss of generality, FIG. 2A and FIG. 2B depict only two layers; however, the methods can easily be extended to systems that support multiple enhancement layers.

10

[0034] As depicted in FIG. 2A, both H.264 encoder 110 and HEVC encoder 120 comprise intra prediction, inter prediction, forward transform and quantization (FT), inverse transforms and quantization (IFT), entropy coding (EC), deblocking filters (DF), and Decoded Picture Buffers (DPB). In addition, an HEVC encoder includes also a Sample Adaptive Offset (SAO) block. In an embodiment, as will be explained later on, RPU 115 may access BL data either before the deblocking filter (DF) or from the DPB. Similarly, in a multi-standard decoder (see FIG. 2B), decoder RPU 135 may also access BL data either before the deblocking filter (DF) or from the DPB.

15

20

[0035] In scalable video coding, the term “multi-loop solution” denotes a layered decoder where pictures in an enhancement layer are decoded based on reference pictures extracted by both the same layer and other sub-layers. The pictures of the base/reference layers are reconstructed and stored in the Decoded Picture Buffer (DPB). These base layer pictures, called inter-layer reference pictures, can serve as additional reference pictures, in decoding the enhancement layer. The enhancement layer then has the options to use either temporal reference pictures or inter-layer reference pictures. In general, inter-layer prediction helps to improve the EL coding efficiency in a scalable system. Since the AVC and HEVC are two different coding standards and they use different encoding processes, additional inter-layer processing may be required to guarantee that AVC-coded pictures are considered valid HEVC reference pictures. In an embodiment, such processing may be performed by RPU 115, as it will be explained next for various cases of interest. For coding standard scalability, the use of RPU 115 aims to resolve the differences or conflicts arising

25

30

from using two different standards, both at a high syntax level and the coding tools level.

Picture Order Count (POC)

- [0036] HEVC and AVC have several differences at the high level syntax. In addition, the same syntax may have a different meaning in each standard. The RPU can work as a high-level syntax “translator” between the base layer and the enhancement layer. One such example is the syntax related to Picture Order Count (POC). In inter-layer prediction, it is important to synchronize the inter-layer reference pictures from the base layer with the pictures being encoded in the enhancement layer. Such synchronization is even more important when the base layer and the enhancement layers use different picture coding structures. For both the AVC and HEVC standards, the term Picture Order Count (POC) is used to indicate the display order of the coded pictures. However, in AVC, there are three methods to signal POC information (indicated by the variable **pic\_order\_cnt\_type**), while in HEVC, only one method is allowed, which is the same as **pic\_order\_cnt\_type == 0** in the AVC case. In an embodiment, when **pic\_order\_cnt\_type** is not equal to 0 in an AVC bitstream, then the RPU (135) will need to translate it into a POC value that conforms to the HEVC syntax. In an embodiment, an encoder RPU (115) may signal additional POC-related data by using a new **pic\_order\_cnt\_lsb** variable, as shown in Table 1. In another embodiment, the encoder RPU may simply force the base layer AVC encoder to only use **pic\_order\_cnt\_type == 0**.

**Table 1 POC syntax**

|                          |                   |
|--------------------------|-------------------|
| POC() {                  | <b>Descriptor</b> |
| <b>pic_order_cnt_lsb</b> | u(1)              |
| }                        |                   |

- [0037] In Table 1, **pic\_order\_cnt\_lsb** specifies the picture order count modulo **MaxPicOrderCntLsb** for the current inter-layer reference picture. The length of the **pic\_order\_cnt\_lsb** syntax element is  $\log_2 \text{max\_pic\_order\_cnt\_lsb\_minus4} + 4$  bits. The value of the **pic\_order\_cnt\_lsb** shall be in the range of 0 to **MaxPicOrderCntLsb – 1**, inclusive. When **pic\_order\_cnt\_lsb** is not present, **pic\_order\_cnt\_lsb** is inferred to be equal to 0.

## Cropping Window

- [0038] In AVC coding, the picture resolution must be a multiple of 16. In HEVC, the resolution can be a multiple of 8. When processing an inter-layer reference picture in the RPU, a cropping window might be used to get rid of padded pixels in AVC. If the base layer and the enhancement layer have different spatial resolution (e.g., a base layer is 1920 x 1080 and the enhancement layer is 4K), or if the picture aspect ratios (PAR) are different (say, 16:9 PAR for the enhancement layer and 4:3 PAR for the base layer), the image has to be cropped and may be resized accordingly. An example of cropping window related RPU syntax is shown in Table 2.

10

**Table 2: Picture Cropping Syntax**

| <code>pic_cropping( ) {</code>         | <b>Descriptor</b> |
|----------------------------------------|-------------------|
| <b>pic_cropping_flag</b>               | u(1)              |
| <code>if( pic_cropping_flag ) {</code> |                   |
| <b>pic_crop_left_offset</b>            | ue(v)             |
| <b>pic_crop_right_offset</b>           | ue(v)             |
| <b>pic_crop_top_offset</b>             | ue(v)             |
| <b>pic_crop_bottom_offset</b>          | ue(v)             |
| <code>}</code>                         |                   |
| <code>}</code>                         |                   |

- [0039] In Table 2, **pic\_cropping\_flag** equal to 1 indicates that the picture cropping offset parameters follow next. If **pic\_cropping\_flag** = 0, then the picture cropping offset parameters are not present and no cropping is required.

- [0040] **pic\_crop\_left\_offset**, **pic\_crop\_right\_offset**, **pic\_crop\_top\_offset**, and **pic\_crop\_bottom\_offset** specify the number of samples in the pictures of the coded video sequence that are input to the RPU decoding process, in terms of a rectangular region specified in picture coordinates for RPU input.

- [0041] Note that since the RPU process is performed for each inter-layer reference, the cropping window parameters can change on a frame-by-frame basis. Adaptive region-of-interest based video retargeting is thus supported using the pan-(zoom)-scan approach.

- [0042] FIG. 3 depicts an example of layered coding, where an HD (e.g., 1920x1080) base layer is coded using H.264 and provides a picture that can be decoded by all legacy HD decoders. A lower-resolution (e.g., 640x480) enhancement

layer may be used to provide optional support for a “zoom” feature. The EL layer has a smaller resolution than the BL, but may be encoded in HEVC to reduce the overall bit rate. Inter-layer coding, as described herein, may further improve the coding efficiency of this EL layer.

5

#### In-loop Deblocking Filter

**[0043]** Both AVC and HEVC employ a deblocking filter (DF) in the coding and decoding processes. The deblocking filter is intended to reduce the blocking artifacts due to the block based coding. But their designs in each standard are quite different.

10 In AVC, the deblocking filter is applied on a 4x4 sample grid basis, but in HEVC, the deblocking filter is only applied to the edges which are aligned on an 8x8 sample grid. In HEVC, the strength of the deblocking filter is controlled by the values of several syntax elements similar to AVC, but AVC supports five strengths while HEVC supports only three strengths. In HEVC, there are less cases of filtering compared to  
15 AVC. For example, for luma, one of three cases is chosen: no filtering, strong filtering and weak filtering. For chroma, there are only two cases: no filtering and normal filtering. To align the deblocking filter operations between the base layer reference picture and a temporal reference picture from the enhancement layer, several approaches can be applied.

20 **[0044]** In one embodiment, the reference picture without AVC deblocking may be accessed directly by the RPU, with no further post-processing. In another embodiment, the RPU may apply the HEVC deblocking filter to the inter-layer reference picture. The filter decision in HEVC is based on the value of several syntax elements, such as transform coefficients, reference index, and motion vectors. It can  
25 be really complicated if the RPU needs to analyze all the information to make a filter decision. Instead, one can explicitly signal the filter index on a 8x8 block level, CU (Coding Unit) level, LCU/CTU (Largest Coding Unit or Coded Tree Unit) level, multiple of LCU level, slice level or picture level. One can signal luma and chroma filter indexes separately or they can share the same syntax. Table 3 shows an example  
30 of how the deblocking filter decision could be indicated as part of an RPU data stream.

**Table 3: Deblocking filter syntax**

|                      |                   |
|----------------------|-------------------|
| deblocking(rx, ry) { | <b>Descriptor</b> |
| <b>filter_idx</b>    | ue(v)             |
| }                    |                   |

[0045] In Table 3, **filter\_idx** specifies the filter index for luma and chroma components. For luma, filter\_idx equal to 0 specifies no filtering, filter\_idx equal to 1 specifies weak filtering, and filter\_idx equal to 2 specifies strong filtering. For chroma, filter\_idx equal to 0 or 1 specifies no filtering, and filter\_idx equal to 2 specifies normal filtering.

#### Sample Adaptive Offset (SAO)

[0046] SAO is a process which modifies, through a look-up table, the samples after the deblocking filter (DF). As depicted in FIG. 2A and FIG. 2B, it is only part of the HEVC standard. The goal of SAO is to better reconstruct the original signal amplitudes by using a look-up table that is described by a few additional parameters that can be determined by histogram analysis at the encoder side. In one embodiment, the RPU can process the deblocking/non-deblocking inter-layer reference picture from the AVC base layer using the exact SAO process as described in HEVC. The signaling can be region based, adapted by CTU (LCU) level, multiple of LCU levels, a slice level, or a picture level. Table 4 shows an example syntax for communicating SAO parameters. In Table 4, the notation syntax is the same as the one described in the HEVC specification.

**Table 4: Sample Adaptive Offset Syntax**

| sao( rx, ry ){                                     | Descriptor |
|----------------------------------------------------|------------|
| if( rx > 0 ) {                                     |            |
| <b>sao_merge_left_flag</b>                         | ue(v)      |
| }                                                  |            |
| if( ry > 0 && !sao_merge_left_flag ) {             |            |
| <b>sao_merge_up_flag</b>                           | ue(v)      |
| }                                                  |            |
| if( !sao_merge_up_flag && !sao_merge_left_flag ) { |            |
| for( cIdx = 0; cIdx < 3; cIdx++ ) {                |            |
| if( ( slice_sao_luma_flag && cIdx == 0 )           |            |
| ( slice_sao_chroma_flag && cIdx > 0 ) ) {          |            |
| if( cIdx == 0 )                                    |            |
| <b>sao_type_idx_luma</b>                           | ue(v)      |
| if( cIdx == 1 )                                    |            |
| <b>sao_type_idx_chroma</b>                         | ue(v)      |
| if( SaoTypeIdx[ cIdx ][ rx ][ ry ] != 0 ) {        |            |
| for( i = 0; i < 4; i++ )                           |            |
| <b>sao_offset_abs</b> [ cIdx ][ rx ][ ry ][ i ]    | ue(v)      |
| if( SaoTypeIdx[ cIdx ][ rx ][ ry ] == 1 ) {        |            |
| for( i = 0; i < 4; i++ ) {                         |            |
| if( sao_offset_abs[ cIdx ][ rx ][ ry ][ i ] != 0 ) |            |
| <b>sao_offset_sign</b> [ cIdx ][ rx ][ ry ][ i ]   | ae(v)      |
| <b>sao_band_position</b> [ cIdx ][ rx ][ ry ]      | ae(v)      |
| } else {                                           |            |
| if( cIdx == 0 )                                    |            |
| <b>sao_eo_class_luma</b>                           | ae(v)      |
| if( cIdx == 1 )                                    |            |
| <b>sao_eo_class_chroma</b>                         | ae(v)      |
| }                                                  |            |
| }                                                  |            |
| }                                                  |            |
| }                                                  |            |
| }                                                  |            |
| }                                                  |            |

## Adaptive Loop Filter (ALF)

[0047] During the development of HEVC, an adaptive loop filter (ALF) was also  
5 evaluated as a processing block following SAO; however, ALF is not part of the first

version of HEVC. Since ALF processing can improve inter-layer coding, if implemented by a future encoder, it is another processing step that could be implemented by the RPU as well. The adaptation of ALF can be region based, adapted by a CTU (LCU) level, multiple of LCU levels, a slice level, or a picture level. An example of ALF parameters is described by `alf_picture_info()` in, “*High efficiency video coding (HEVC) text specification draft 7*,” by B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-I1003, May 2012, which is incorporated herein by reference in its entirety.

10

#### Interlaced and Progressive Scanning

**[0048]** AVC supports coding tools for both progressive and interlaced content. For interlaced sequences, it allows both frame coding and field coding. In HEVC, no explicit coding tools are present to support the use of interlaced scanning. HEVC provides only metadata syntax (Field Indication SEI message syntax and VUI) to allow an encoder to indicate how interlaced content was coded. The following scenarios are considered.

15

Scenario 1: Both the base layer and the enhancement layer are interlaced

20

**[0049]** For this scenario, several methods can be considered. In a first embodiment, the encoder may be constrained to change the base layer encoding in a frame or field mode only on a per sequence basis. The enhancement layer will follow the coding decision from the base layer. That is, if the AVC base layer uses field coding in one sequence, the HEVC enhancement layer will use field coding in the corresponding sequence too. Similarly, if the AVC base layer uses frame coding in one sequence, the HEVC enhancement layer will use frame coding in the corresponding sequence too. It is noted that for field coding, the vertical resolution signaled in the AVC syntax is the frame height; however, in HEVC, the vertical resolution signaled in the syntax is the field height. Special care must be taken in communicating this information in the bit stream, especially if a cropping window is used.

25

30

**[0050]** In another embodiment, the AVC encoder may use picture-level adaptive frame or field coding, while the HEVC encoder performs sequence-level adaptive

frame or field coding. In both cases, the RPU can process inter-layer reference pictures in one of the following ways: a) The RPU may process the inter-layer reference picture as fields, regardless of the frame or field coding decision in the AVC base layer, or b) the RPU may adapt the processing of the inter-layer reference pictures based on the frame/field coding decision in the AVC base layer. That is, if the AVC base layer is frame-coded, the RPU will process the inter-layer reference picture as a frame, otherwise, it will process the inter-layer reference picture as fields.

**[0051]** FIG. 4 depicts an example of Scenario 1. The notation  $D_i$  or  $D_p$  denotes frame rate and whether the format is interlaced or progressive. Thus,  $D_i$  denotes  $D$  interlaced frames per second (or  $2D$  fields per second) and  $D_p$  denotes  $D$  progressive frames per second. In this example, the base layer comprises a standard-definition (SD) 720x480, 30i sequence coded using AVC. The enhancement layer is a high-definition (HD) 1920x1080, 60i sequence, coded using HEVC. This example incorporates codec scalability, temporal scalability, and spatial scalability. Temporal scalability is handled by the enhancement layer HEVC decoder using a hierarchical structure with temporal prediction only (this mode is supported by HEVC in a single-layer). Spatial scalability is handled by the RPU, which adjusts and synchronizes slices of the inter-layer reference field/frame with its corresponding field/frame slices in the enhancement layer.

Scenario 2: The base layer is interlaced and the enhancement layer is progressive

**[0052]** In this scenario, the AVC base layer is an interlaced sequence and the HEVC enhancement layer is a progressive sequence. FIG. 5A depicts an example embodiment wherein an input 4K 120p signal (502) is encoded as three layers: a 1080 30i BL stream (532), a first enhancement layer (EL0) stream (537), coded as 1080 60p, and a second enhancement layer stream (EL1) (517), coded as 4K 120p. The BL and EL0 signals are coded using an H.264/AVC encoder while the EL1 signal may be coded using HEVC. On the encoder, starting with a high-resolution, high-frame 4K, 120p signal (502), the encoder applies temporal and spatial down-sampling (510) to generate a progressive 1080 60p signal 512. Using a complementary progressive to deinterlacing technique (520), the encoder may also generate two complimentary, 1080 30i, interlaced signals BL 522-1 and EL0 522-2. As used herein, the term “complementary progressive to deinterlacing technique” denotes a scheme that

generates two interlaced signals from the same progressive input, where both interlaced signals have the same resolution, but one interlaced signal includes the fields from the progressive signal that are not part of the second interlaced signal. For example, if the input signal at time  $T_i$ ,  $i=0, 1, \dots, n$ , is divided into top and bottom interlaced fields (Top- $T_i$ , Bottom- $T_i$ ), then the first interlaced signal may be constructed using (Top- $T_0$ , Bottom- $T_1$ ), (Top- $T_2$ , Bottom- $T_3$ ), etc., while the second interlaced signal may be constructed using the remaining fields, that is: (Top- $T_1$ , Bottom- $T_0$ ), (Top- $T_3$ , Bottom- $T_2$ ), etc.

**[0053]** In this example, the BL signal 522-1 is a backward-compatible interlaced signal that can be decoded by legacy decoders, while the EL0 signal 522-2 represents the complimentary samples from the original progressive signal. For the final picture composition of the full frame-rate, every reconstructed field picture from the BL signal must be combined with a field picture within the same access unit but with the opposite field parity. Encoder 530 may be an AVC encoder that comprises two AVC encoders (530-1 and 530-2) and RPU processor 530-3. Encoder 530 may use interlayer processing to compress signal EL0 using reference frames from both the BL and the EL0 signals. RPU 530-3 may be used to prepare the BL reference frames used by the 530-2 encoder. It may also be used to create progressive signal 537, to be used for the coding of the EL1 signal 502 by EL1 encoder 515.

**[0054]** In an embodiment, an up-sampling process in the RPU (535) is used to convert the 1080 60p output (537) from RPU 530-3 into a 4K 60p signal to be used by HEVC encoder 515 during inter-layer prediction. EL1 signal 502 may be encoded using temporal and spatial scalability to generate a compressed 4K 120p stream 517. Decoders can apply a similar process to either decode a 1080 30i signal, a 1080 60p signal, or a 4K 120p signal.

**[0055]** FIG. 5B depicts another example implementation of an interlaced/progressive system according to an embodiment. This is a two layer system, where a 1080 30i base layer signal (522) is encoded using an AVC encoder (540) to generate a coded BL stream 542, and a 4K 120p enhancement layer signal (502) is encoded using an HEVC encoder (515) to generate a coded EL stream 552. These two streams may be multiplexed to form a coded scalable bit stream 572.

**[0056]** As depicted in FIG. 5B, RPU 560 may comprise two processes: a de-interlacing process, which converts BL 522 to a 1080 60p signal, and an up-sampling

process to convert the 1080 60p signal back to a 4K 60p signal, so the output of the RPU may be used as a reference signal during inter-layer prediction in encoder 515.

Scenario 3: The base layer is progressive and the enhancement layer is interlaced

5

In this scenario, in one embodiment, the RPU may convert the progressive inter-layer reference picture into an interlaced picture. These interlaced pictures can be processed by the RPU as a) always fields, regardless of whether the HEVC encoder uses sequence-based frame or field coding, or as b) fields or frames, depending on the mode used by the HEVC encoder. Table 5 depicts an example syntax that can be used to guide the decoder RPU about the encoder process.

10

**Table 5: Interlace Processing Syntax**

| interlace_process( ) {     | Descriptor |
|----------------------------|------------|
| <b>base_field_seq_flag</b> | u(1)       |
| <b>enh_field_seq_flag</b>  | u(1)       |
| }                          |            |

[0057] In Table 5, **base\_field\_seq\_flag** equal to 1 indicates that the base layer coded video sequence conveys pictures that represent fields. **base\_field\_seq\_flag** equal to 0 indicates that the base layer coded video sequence conveys pictures that represent frames.

15

[0058] **enh\_field\_seq\_flag** equal to 1 indicates that the enhancement layer coded video sequence conveys pictures that represent fields. **enh\_field\_seq\_flag** equal to 0 indicates that the enhancement layer coded video sequence conveys pictures that represent frames.

20

[0059] Table 6 shows how an RPU may process the reference pictures based on the **base\_field\_seq\_flag** or **enh\_field\_seq\_flag** flags.

25

**Table 6: RPU processing for progressive/interlaced scanning sequences**

| <b>base_field_seq_flag</b> | <b>enh_field_seq_flag</b> | <b>RPU processing</b>  |
|----------------------------|---------------------------|------------------------|
| 1                          | 1                         | field                  |
| 1                          | 0                         | De-interlacing + frame |
| 0                          | 1                         | Interlacing + field    |
| 0                          | 0                         | frame                  |

## Signal Encoding Model Scalability

[0060] Gamma-encoding is arguably the most widely used signal encoding model, due to its efficiency for representing standard dynamic range (SDR) images. In recent research for high-dynamic range (HDR) imaging, it was found that for several types of images, other signal encoding models, such as the Perceptual Quantizer (PQ) described in “*Parameter values for UHD TV*”, a submission to SG6 WP 6C, WP6C/USA002, by Craig Todd, or U.S. Provisional patent application with Ser. Number 61/674,503, filed on July 23, 2012, and titled “*Perceptual luminance nonlinearity-based image data exchange across different display capabilities*,” by Jon S. Miller et al., both incorporated herein by reference in their entirety, could represent the data more efficiently. Therefore, it is possible that a scalable system may have one layer of SDR content which is gamma-coded, and another layer of high dynamic range content which is coded using other signal encoding models.

[0061] FIG. 6 depicts an embodiment where RPU 610 (e.g., RPU 115 in FIG. 1) may be set to adjust the signal quantizer of the base layer. Given a BL signal 102 (e.g., 8-bit, SDR video signal, gamma encoded in 4:2:0 Rec. 709), and an EL signal 104 (e.g., 12-bit HDR video signal, PQ encoded in 4:4:4 in P3 color space), processing in RPU 610 may comprise: gamma decoding, other inverse mappings (e.g., color space conversions, bit-depth conversions, chroma sampling, and the like), and SDR to HDR perceptual quantization (PQ). The signal decoding and encoding method (e.g., gamma and PQ), and related parameters, may be part of metadata that are transmitted together with the coded bitstream or they can be part of a future HEVC syntax. Such RPU processing may be combined with other RPU processing related to other types of scalabilities, such as bit-depth, chroma format, and color space scalability. As depicted in FIG. 1, similar RPU processing may also be performed by a decoder RPU during the decoding of the scalable bit stream 127.

[0062] Scalability extension can include several other categories, such as: spatial or SNR scalability, temporal scalability, bit-depth scalability, and chroma resolution scalability. Hence, an RPU can be configured to process inter-layer reference pictures under a variety of coding scenarios. For better encoder-decoder compatibility, encoders may incorporate special RPU-related bit stream syntax to guide the corresponding RPU decoder. The syntax can be updated at a variety of coding levels, including: the slice level, the picture level, the GOP level, the scene level, or at the sequence level. It also can be included in a variety of auxiliary data, such as: the NAL

unit header, Sequence Parameter Set (SPS) and its extension, SubSPS, Picture Parameter Set (PPS), slicer header, SEI message, or a new NAL unit header. Since there may be a lot of RPU-related processing tools, for maximum flexibility and ease of implementation, in one embodiment, we propose to reserve a new NAL unit type

- 5 for the RPU to make it a separate bitstream. Under such an implementation, a separate RPU module is added to the encoder and decoder modules to interact with the base layer and the one or more enhancement layers. Table 7 shows an example of RPU data syntax which includes rpu\_header\_data() (shown in Table 8) and rpu\_payload\_data() (shown in Table 9), in a new NAL unit. In this example, multiple  
10 partitions are enabled to allow region based deblocking and SAO decisions.

**Table 7: RPU data syntax**

| <b>rpu_data () {</b>        | <b>Descriptor</b> |
|-----------------------------|-------------------|
| <b>rpu_header_data()</b>    |                   |
| <b>rpu_payload_data()</b>   |                   |
| <b>rbsp_trailing_bits()</b> |                   |
| <b>}</b>                    |                   |

**Table 8: RPU header data syntax**

| <b>rpu_header_data () {</b>    | <b>Descriptor</b> |
|--------------------------------|-------------------|
| <b>rpu_type</b>                | u(6)              |
| <b>POC()</b>                   |                   |
| <b>pic_cropping()</b>          |                   |
| <b>deblocking_present_flag</b> | u(1)              |
| <b>sao_present_flag</b>        | u(1)              |
| <b>alf_present_flag</b>        | u(1)              |
| <b>if (alf_present_flag)</b>   |                   |
| <b>alf_picture_info()</b>      |                   |
| <b>interlace_process()</b>     |                   |
| <b>num_x_partitions_minus1</b> | ue(v)             |
| <b>num_y_partitions_minus1</b> | ue(v)             |
| <b>}</b>                       |                   |

15

**Table 9: RPU payload data syntax**

| <code>rpu_payload_data () {</code>                                                                 | <b>Descriptor</b> |
|----------------------------------------------------------------------------------------------------|-------------------|
| <code>for (y = 0, y &lt;= num_y_partitions_minus1; y++ ) {</code>                                  |                   |
| <code>for (x = 0; x &lt;= num_x_partitions_minus1; x++ ) {</code>                                  |                   |
| <code>if (deblocking_present_flag)</code>                                                          |                   |
| <code>deblocking()</code>                                                                          |                   |
| <code>if (sao_present_flag)</code>                                                                 |                   |
| <code>sao()</code>                                                                                 |                   |
| <code>/* below is to add other parameters related to upsampling filter,<br/>mapping, etc.*/</code> |                   |
| <code>/* example 1: if (rpu_type == SPATIAL_SCALABILITY) */</code>                                 |                   |
| <code>/* rpu_process_spatial_scalability() */</code>                                               |                   |
| <code>/* example 2: if (rpu_type == BIT_DEPTH_SCALABILITY) */</code>                               |                   |
| <code>/* rpu_process_bit_depth_scalability() */</code>                                             |                   |
| <code>....</code>                                                                                  |                   |
| <code>}</code>                                                                                     |                   |
| <code>}</code>                                                                                     |                   |
| <code>}</code>                                                                                     |                   |

[0063] In Table 8, **rpu\_type** specifies the prediction type purpose for the RPU signal. It can be used to specify different kinds of scalability. For example, rpu\_type equal to 0 may specify spatial scalability, and rpu\_type equal to 1 may specifies bit-depth scalability. In order to combine different scalability modes, one may also use a masking variable, such as **rpu\_mask**. For example, rpu\_mask=0x01 (binary 00000001) may denote that only spatial scalability is enabled. rpu\_mask=0x02 (binary 00000010) may denote that only bit-depth scalability is enabled. rpu\_mask = 0x03 (binary 00000011) may denote that both spatial and bit-depth scalability are enabled.

[0064] **deblocking\_present\_flag** equal to 1 indicates syntax related to deblocking filter is present in the RPU data.

[0065] **sao\_present\_flag** equal to 1 indicates syntax related to SAO is present in the RPU data.

[0066] **alf\_present\_flag** equal to 1 indicates syntax related to ALF filter is present in the RPU data.

[0067] **num\_x\_partitions\_minus1** signals the number of partitions that are used to subdivide the processed picture in the horizontal dimension in RPU.

[0068] **num\_y\_partitions\_minus1** signals the number of partitions that are used to subdivide the processed picture in the vertical dimension in RPU.

[0069] In another embodiment, instead of using POC to synchronize the base layer and enhancement layer pictures, the RPU syntax is signaled at the picture level, so multiple pictures can reuse the same RPU syntax, which result in lower bit overhead and possibly reducing processing overhead in some implementations. Under this implementation, the **rpuid** will be added into the RPU syntax. In **slice\_header()**, it will always refer to **rpuid** to synchronize RPU syntax with the current slice, where the **rpuid** variable identifies the **rpuidata()** that is referred to in the slice header.

[0070] FIG. 7 depicts an example encoding process according to an embodiment. Given a series of pictures (or frames), the encoder encodes a base layer with a BL encoder using a first compression standard (e.g., AVC) (715). Next (720, 725), as depicted in FIG. 2A and 2B, RPU process 115, may access base layer pictures either before or after the deblocking filter (DF) The decision can be made based on RD (rate-distortion) optimization or the processing that RPU performs. For example, if RPU performs up-sampling, which may also be used in deblocking the block boundaries, then the RPU may just use the decoded base layer before the deblocking filter, and the up-sampling process may retain more details. RPU 115 may determine the RPU processing parameters based on the BL and EL coding parameters. If needed, the RPU process may also access data from the EL input. Then, in step 730, the RPU processes the inter-layer reference pictures according to the determined RPU process parameters. The generated inter-layer pictures (735) may now be used by the EL encoder using a second compression standard (e.g., an HEVC encoder) to compress the enhancement layer signal.

[0071] FIG. 8 depicts an example decoding process according to an embodiment. First (810), the decoder parses the high-level syntax of the input bitstream to extract sequence parameters and RPU-related information. Next (820), it decodes the base layer with a BL decoder according to the first compression standard (e.g., an AVC decoder). After decoding the RPU-process related parameters (825), the RPU process generates inter-layer reference pictures according to these parameters (steps 830 and 835). Finally, the decoder decodes the enhancement layer using an EL decoder that complies with the second compression standard (e.g., an HEVC decoder) (840).

[0072] Given the example RPU parameters defined in Tables 1-9, FIG. 9 depicts an example decoding RPU process according to an embodiment. First (910), the

decoder extracts from the bitstream syntax the high-level RPU-related data, such as RPU type (e.g., **rpu\_type** in Table 8), POC(), and pic\_cropping(). The term “RPU type” refers to RPU-related sub-processes that need to be considered, such as: coding-standard scalability, spatial scalability, bit-depth scalability, and the like, as discussed earlier. Given a BL frame, cropping, and ALF-related operations may be processed first (e.g., 915, 925). Next, after extracting the required interlaced or deinterlaced mode (930), for each partition, the RPU performs deblocking and SAO-related operations (e.g., 935, 940). If additional RPU processing needs to be performed (945), then the RPU decodes the appropriate parameters (950) and then performs operations according to these parameters. At the end of this process, a sequence of inter-layer frames is available to the EL decoder to decode the EL stream.

#### EXAMPLE COMPUTER SYSTEM IMPLEMENTATION

[0073] Embodiments of the present invention may be implemented with a computer system, systems configured in electronic circuitry and components, an integrated circuit (IC) device such as a microcontroller, a field programmable gate array (FPGA), or another configurable or programmable logic device (PLD), a discrete time or digital signal processor (DSP), an application specific IC (ASIC), and/or apparatus that includes one or more of such systems, devices or components. The computer and/or IC may perform, control or execute instructions relating to RPU processing, such as those described herein. The computer and/or IC may compute any of a variety of parameters or values that relate to RPU processing as described herein. The RPU-related embodiments may be implemented in hardware, software, firmware and various combinations thereof.

[0074] Certain implementations of the invention comprise computer processors which execute software instructions which cause the processors to perform a method of the invention. For example, one or more processors in a display, an encoder, a set top box, a transcoder or the like may implement methods RPU processing as described above by executing software instructions in a program memory accessible to the processors. The invention may also be provided in the form of a program product. The program product may comprise any medium which carries a set of computer-readable signals comprising instructions which, when executed by a data processor, cause the data processor to execute a method of the invention. Program products according to the invention may be in any of a wide variety of forms. The

program product may comprise, for example, physical media such as magnetic data storage media including floppy diskettes, hard disk drives, optical data storage media including CD ROMs, DVDs, electronic data storage media including ROMs, flash RAM, or the like. The computer-readable signals on the program product may

5 optionally be compressed or encrypted.

[0075] Where a component (e.g. a software module, processor, assembly, device, circuit, etc.) is referred to above, unless otherwise indicated, reference to that component (including a reference to a "means") should be interpreted as including as equivalents of that component any component which performs the function of the described component (e.g., that is functionally equivalent), including components  
10 which are not structurally equivalent to the disclosed structure which performs the function in the illustrated example embodiments of the invention.

#### EQUIVALENTS, EXTENSIONS, ALTERNATIVES AND MISCELLANEOUS

15 [0076] Example embodiments that relate to RPU processing and standards-based codec scalability are thus described. In the foregoing specification, embodiments of the present invention have been described with reference to numerous specific details that may vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and is intended by the applicants to be the  
20 invention, is the set as recited in claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation, element, property, feature, advantage or attribute that is not expressly recited in a claim should  
25 limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

---

## CLAIMS

1. A method for encoding input data in a coding-standard layered stream, the method comprising:

- 5 encoding a base layer (BL) signal into a coded BL stream using a BL encoder,  
the BL encoder being compliant to a first coding standard;  
in response to the BL signal and an enhancement layer (EL) signal,  
determining RPU processing parameters using a reference processing unit (RPU);  
in response to the RPU processing parameters and the BL signal, generating  
an inter-layer reference signal using the RPU;  
10 encoding the EL signal using an EL encoder into a coded EL stream, the EL  
encoder being compliant to a second coding standard, the encoding of the EL signal  
being based at least in part on the inter-layer reference signal.

2. The method of claim 1, wherein the first coding standard comprises encoding  
15 according to the H.264 standard and the second coding standard comprises encoding  
according to the HEVC standard.

3. The method of claim 1, wherein determining the RPU processing parameters  
further comprises resolving with the RPU one or more differences between the first  
20 and the second coding standard, wherein the differences are related to high-level  
syntax or coding tools.

4. The method of claim 3, wherein the RPU processing parameters comprise one or  
more parameters related to picture order count (POC), cropping windows, in-loop  
25 deblocking filter, sample adaptive offset (SAO), adaptive loop filter (ALF), interlaced  
and progressive scanning, or signal encoding model.

5. The method of claim 4, wherein the RPU processing parameters related to cropping  
window comprise a variable that indicates that one or more picture cropping offset  
30 parameters follow next, followed by the one or more picture cropping offset  
parameters.

6. The method of claim 4, wherein the RPU processing parameters related to the in-loop deblocking filter comprise a variable that indicates whether to perform no filtering, weak filtering, or strong filtering.
- 5 7. The method of claim 3, where the RPU resolves differences in the syntax of a picture order count (POC) variable and translates the syntax of the POC variable from a first value defined in the coded BL stream to a second value compliant with the second coding standard.
- 10 8. The method of claim 3, wherein the RPU resolves differences in the coding tools related to coding interlaced and progressive pictures.
9. The method of claim 8, wherein the RPU processing parameters related to coding interlaced and progressive pictures comprise a first variable that indicates whether the  
15 base layer signal is an interlaced or progressive signal and a second variable that indicates whether the EL signal is an interlaced or progressive signal.
10. The method of claim 3, wherein the RPU resolves differences to the coding tools related to inner-loop deblocking filtering.
- 20 11. A method for decoding a coding-standard layered stream, the method comprising:  
receiving a coded BL stream, a coded EL stream, and an RPU data stream;  
decoding the coded BL stream using a BL decoder to generate a decoded BL signal, the BL decoder being compliant to a first coding standard;  
25 decoding the RPU data stream with an RPU to determine RPU process parameters;  
in response to the RPU processing parameters and the BL signal, generating an inter-layer reference signal using the RPU;  
decoding the coded EL stream using an EL decoder to generate a decoded EL  
30 signal, the EL decoder being compliant to a second coding standard, the decoding of the EL signal being based at least in part on the inter-layer reference signal.

12. The method of claim 11, wherein the first coding standard comprises decoding according to the H.264 standard and the second coding standard comprises decoding according to the HEVC standard.

5 13. The method of claim 11, wherein the RPU processing parameters comprise one or more parameters related to picture order count (POC), cropping windows, in-loop deblocking filter, sample adaptive offset (SAO), adaptive loop filter (ALF), interlaced and progressive scanning, or signal encoding model.

10 14. The method of claim 11, wherein the RPU processing parameters related to cropping window comprise a variable that indicates that picture cropping offset parameters follow next, followed by one or more variables determining a rectangular region.

15 15. The method of claim 11, wherein the RPU processing parameters related to the in-loop deblocking filter comprise a variable that indicates whether to perform no filtering, weak filtering, or strong filtering.

16. The method of claim 11, wherein the RPU processing parameters related to  
20 interlaced and progressive scanning comprise a first variable that indicates whether the base layer signal is an interlaced or progressive signal and a second variable that indicates whether the EL signal is an interlaced or progressive signal.

17. An apparatus comprising a processor and configured to perform any one of the  
25 methods recited in Claims 1 or 11.

18. A non-transitory computer-readable storage medium having stored thereon computer-executable instructions for executing a method in accordance with any of the Claims 1 or 11.

30