Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11
11th Meeting: Shanghai, CN, 10-19 Oct., 2012

Document: JCTVC-K0041

Title: **Description of the scalable video coding technology proposal by Canon Research Centre France**

Status: Input Document to JCT-VC

Purpose: Proposal

Author(s) or Contact(s):
Sébastien LASSERRE
Fabrice LE LEANNEC
Jonathan TAQUET
Naël OUEDRAOGO
Stéphane PAUTET
Christophe GISQUET
Guillaume LAROCHE
Tangi POIRIER
Yann VERDAVAINE
Edouard FRANCOIS
Eric NASSOR

Canon Research Centre France
Rue de la Touche Lambert
35510 CESSON-SEVIGNE, FRANCE

Tel: +33(0)299876800
Email: sebastien.lasserre@crf.canon.fr
fabrice.leleannec@crf.canon.fr
eric.nassor@crf.canon.fr

Source: Canon Research Centre France

Abstract

This document describes Canon's response to the Joint Call for Proposals for the scalable video coding extensions of HEVC. The proposed codec responds to the HEVC base layer categories of the CfP: Spatial, Intra-only and SNR scalability.
This document describes the coding tools added to HEVC coding tools for scalability: an inter-layer intra coding technology for intra frames and Inter-layer prediction tools for Inter frames.
In terms of coding efficiency, it is reported that the proposed codec shows scalable performance with a mean overhead of about 8.5% in All Intra configurations, 6.7% in Random Access spatial configurations and 3.1% in random Access SNR configuration compared to the monocast (single layer) HM6.1. Overall, a gain of 44.3% over the simulcast enhancement layer is obtained with the proposed scalability layer coding system. This corresponds to a gain of roughly 29% compared to the simulcast HM6.1, in the coding of the base plus the enhancement layer.

Contents

1 Introduction

This contribution describes Canon's proposal in response to the Joint Call for Proposals for the scalable video coding extensions of HEVC [2]. Canon's proposal, called the Canon Scalable Codec (CSC), is based on the HM6.1 reference software and enables spatial and SNR scalabilities over an HEVC base layer.

The Canon proposal responds to the following configurations of the CfP:

- ☒ Category 1 (HEVC base layer) spatial scalability
- ☒ Category 1 (HEVC base layer) intra-only spatial scalability
- ☒ Category 1 (HEVC base layer) SNR scalability
- ☐ Category 2 (AVC base layer) spatial scalability

- ☒ The proposal obeys the constraints under section 5 of the CfP (if box is not ticked, explain cases where constraints are violated)

Notes:

- As in HEVC, the quantization settings for a frame in the enhancement layer are changing according to its position in the GOP (see section 2.4.2.1).
- In the intra-only solution, the quantization technology does not make use of QPs, and consists of an automatic optimal rate balancing among the different enhancement layer frames and block sizes based on the signal statistics. In addition the balance between luma and chroma has been adjusted in order to reach a similar balance as a pure HEVC-based scalable codec. This is detailed in sections 2.1.1 and 2.3.

In terms of coding efficiency, it is reported that the proposed codec shows scalable performance with a mean overhead of about 8.5% in All Intra configurations, 6.7% in Random Access spatial configurations and 3.1% in random Access SNR configuration compared to the monocast (single layer) HM6.1. Overall, a gain of 44.3% over simulcast enhancement layer is obtained with the proposed scalability layer coding system. This corresponds to a gain of roughly 29% compared to the simulcast HM6.1, in the coding of the base plus the enhancement layer.

The main characteristics of the CSC codec design are the following. Several video coding tools have been added to the HEVC coding tools to improve the coding efficiency for scalable coding: an Inter-layer intra coding technology for intra frames and Inter-layer prediction tools for Inter frames.

The Inter-layer intra coding tool is a low complexity coding tool aiming at reducing complexity compared to standard intra coding systems, while providing high coding efficiency. Complexity reduction comes from the use of only one coding mode in enhancement intra pictures, which is an Inter-layer intra prediction. The good coding efficiency performance of this tool is obtained through the statistical modeling of DCT channels to encode and the design of optimal rate-distortion quantizers. These quantizers that are pre-computed off-line are coupled with a bit allocation process among DCT channels, components and intra frames. This tool is used in the all-intra configuration and for the intra-slices of the random-access configuration.
Moreover, this intra coding tool only uses the decoded pixels from the base layer for Inter-layer prediction. It is base layer agnostic which means that it does not depend on any standard-specific decoded element of the base layer and could be used as is over an AVC base layer.
In addition, the removal of spatial prediction, combined with a context-free and non-adaptive arithmetic coding, provide spatial random access feature, together with a high degree of possible parallelism (decoder and encoder).

With respect to the Random Access coding structure, the above enhancement intra picture coding tool is integrated with enhancement Inter pictures coding, through a bit allocation process between IDR and B pictures. Enhancement Inter pictures are coded in an HEVC-like way, with additional coding modes for Inter-layer prediction. Several Inter layer prediction tools are added to already specified HEVC prediction tools. Intra Coding Units employ Base Mode, Intra BL, and difference intra coding proposed by Vidyo [4]. HEVC spatial intra prediction has been removed for the enhancement layer. Inter coding modes of HEVC draft 6 are re-used and have been extended with a new mode presented in this contribution called the Generalized Residual Inter-Layer Prediction (GRILP). AMVP and Merge derivations processes are extended to provide better Inter-layer prediction of motion information.

In this contribution, in-loop filtering of the reconstructed enhancement pictures includes de-blocking filtering, improved SAO and ALF.

Finally, 10-bit precision is used for internal processing in the CSC codec, so as to provide support for increased bit-depth. Performances are reported both with 8-bit and 10-bit measured PSNR's. For information, results of SNR scalability for intra-only scenario are also provided.

The CSC software is based on the source code of HM6.1 initially modified by Vidyo to provide a scalable extension to HEVC [4, 17]. The classical structure of the software enables easy integration and test of new tools. Every required bit-streams for the CFP are provided with HEVC base layer, and are decoded by the CSC decoder synchronously with the encoder. No bug is known so far.
A base layer extractor is provided together with the codec. The CSC bit-stream is fully backward compliant with HM6.1, which means extracted base layers are decodable with HM6.1 and the CSC codec behaves the same way as HM6.1 in non-scalable configuration. CSC source code could thus be used as reference software for the collaborative design of scalable HEVC extension.

2 Algorithm description

2.1 Overview of the Canon Scalable Codec (CSC)

2.1.1 All-Intra coding: coding structure

This section introduces the CSC Intra coding structure developed for the scalable HEVC extension. This structure is proposed for the All-Intra configuration, but also applies to the intra slices of the Random Access configuration. A complete description of the CSC Intra coding process is given in section 2.3.
Figure 1 depicts the coding structure employed in the proposed scalable codec in All-Intra configuration. This coding structure simply consists in a series of Intra pictures in the HEVC base layer. Then, an enhancement Intra picture is being encoded on top of each Intra base picture.

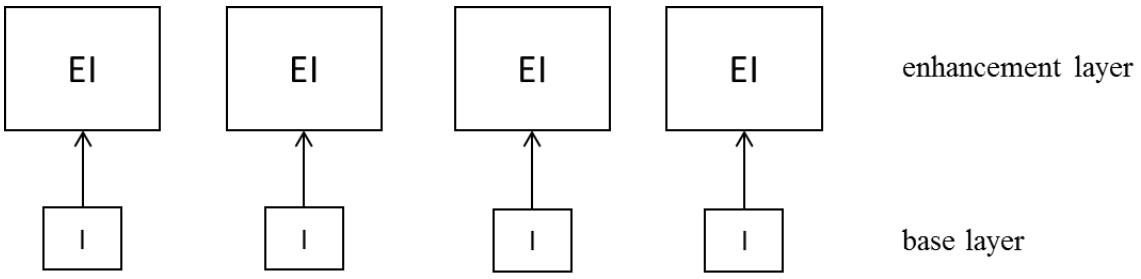

Figure 1: Coding structure in All-Intra configuration

2.1.1.1 Intra coding technology principles

The principle of the proposed intra coding technology is to directly encode the Inter-layer residual, resulting from the difference between the upsampled reconstructed base layer and the source enhancement layer. No Inter-dependency between CUs is used (no spatial prediction, no context dependencies), which both enables a low-complexity decoding solution and offers full spatial random access capabilities. The basis of the proposed technology is to consider the Transform coefficients (using the HEVC transforms) of the Inter-layer residual as independent channels each modeled by **Generalized Gaussian Distributions** (GGD). This modeling enables to derive optimal quantizers, as well as optimal bit-rate balances between these different channels. This balancing is also optimized between the different transforms sizes, color components and pictures of the sequence, using the concept of merit directly related to the rate-distortion slope.
The two next sub-sections give an overview of the encoding and decoding processes. A more detailed description of the different steps involved in these processes is given in section 2.3.

2.1.1.2 Intra-only Encoder

The coding process of each enhancement Intra picture implies two main steps, illustrated on Figure 2. First, the reconstructed base picture is up-sampled towards the spatial resolution of the enhancement layer, in case of spatial scalability. DCT-IF interpolation filters are used in this up-sampling step. They are described in section 2.2. Second, the texture residual picture between the original picture and the up-sampled base picture is computed. This residual image is then encoded according to the Intra texture coding process corresponding to the upper part of Figure 2.

The input to the intra pictures encoder consists in a set of DCT blocks of the residual image. These blocks actually correspond to Transform Units. The blocks are classified into so-called **block types** which consist in a given block size and block label. The block type is decided through an initial guess based on the (morphological) activity analysis of the residual picture (described in section 2.3.1) and this guess is later refined thanks to a **competition process between block types** (section 2.3.9).
Then, for each block type, statistics (taken on all the blocks of the same block type) of the DCT coefficients are computed. By using the computed statistics, **Generalized Gaussian Distribution** (GGD) is fitted to each DCT coefficient channel of each block type (section 2.3.2).
Following its GGD, a quantizer (taken from a pool of quantizers) is assigned to each DCT channel through a process of balancing the merits of encoding (introduced later) between channels of each block type, between block types and between frames and colour components (sections 2.3.5, 2.3.6 and 2.3.7). Quantizers are **non-uniform scalar quantizers** defined by a set of quantization intervals and associated de-quantized sample values. The pool of quantizers is available on both the encoder and on the decoder sides; this pool is made of several quantizers (corresponding to various rates of encoding) that are pre-computed off-line, through the Chou-Lookabaugh-Gray rate distortion optimization process [6]. This allows being optimal (in the rate-distortion sense) on GGD channels (sections 2.3.3 and 2.3.4).
The encoding of the statistics in the bitstream is described in section 2.3.10.

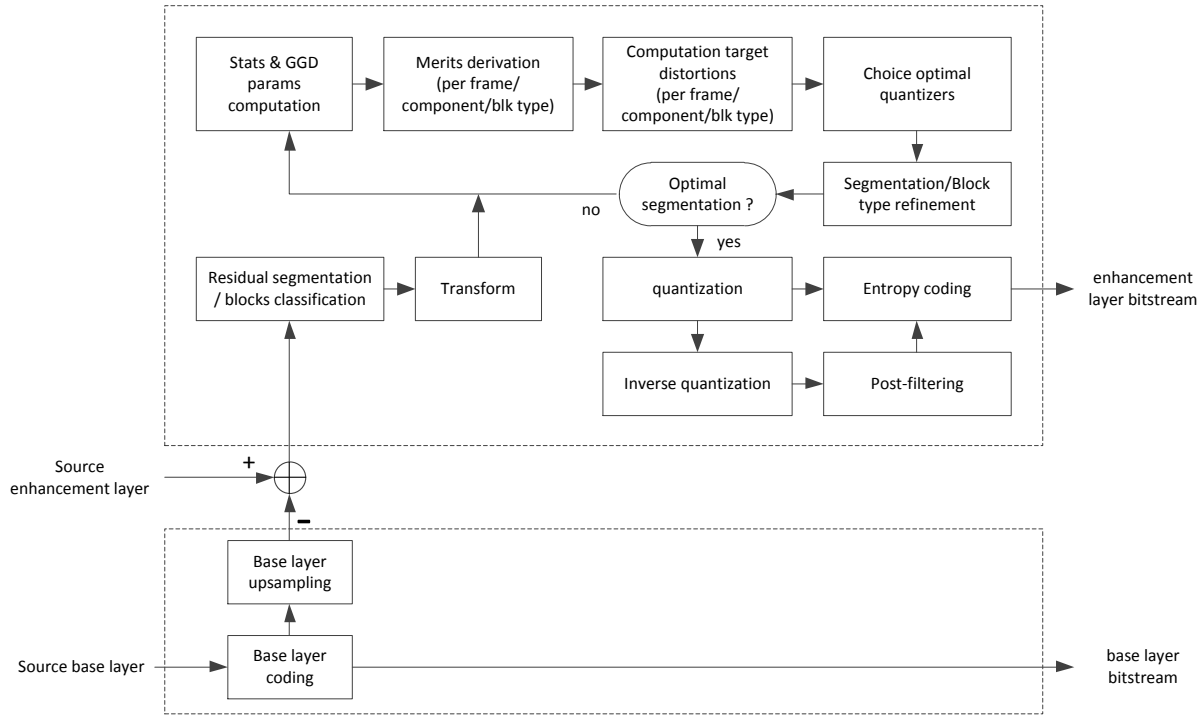

Figure 2: Synoptic of intra-only encoding process

Note that the proposed Intra enhancement picture coding process is of low-complexity, i.e. it does not involves any coding mode decision step as in standard video coding systems and any Inter-dependency between CUs, allowing full spatial random access capabilities. In the CSC only one coding mode is involved in enhancement Intra pictures, which corresponds to so-called **Inter-layer intra prediction**.

2.1.1.3 Intra-only Decoder

The overall enhancement Intra picture decoding process is illustrated in Figure 3. The input bit-stream consists in the HEVC-coded base layer and the enhancement layer comprising coded enhancement Intra pictures. First, the base layer is decoded, which provides a reconstructed base picture. The reconstructed base picture is then up-sampled to reach the enhancement layer resolution. Then, the enhancement layer is decoded as follows. In the entropy decoding, the decoder control parameters (Y, U and V merits of the slice, GGD parameter of each DCT channel) and the block types segmentation are decoded (sections 2.3.1.3, 2.3.10). From the decoded Y, U and V merits, the merits of each block type are derived. They are used to compute the target distortions, according to the decoded GGD parameters, which subsequently are used to select the quantizers for each DCT channel. It is then possible to entropy decode the texture (section 2.3.8). Then the quantized values are de-quantized to generate the transformed

quantized residual. Finally the inverse transform is applied, and the resulting residual is added to the corresponding up-sampled reconstructed base layer slice. The resulting slice is then processed by the post-filtering (section 2.3.11).

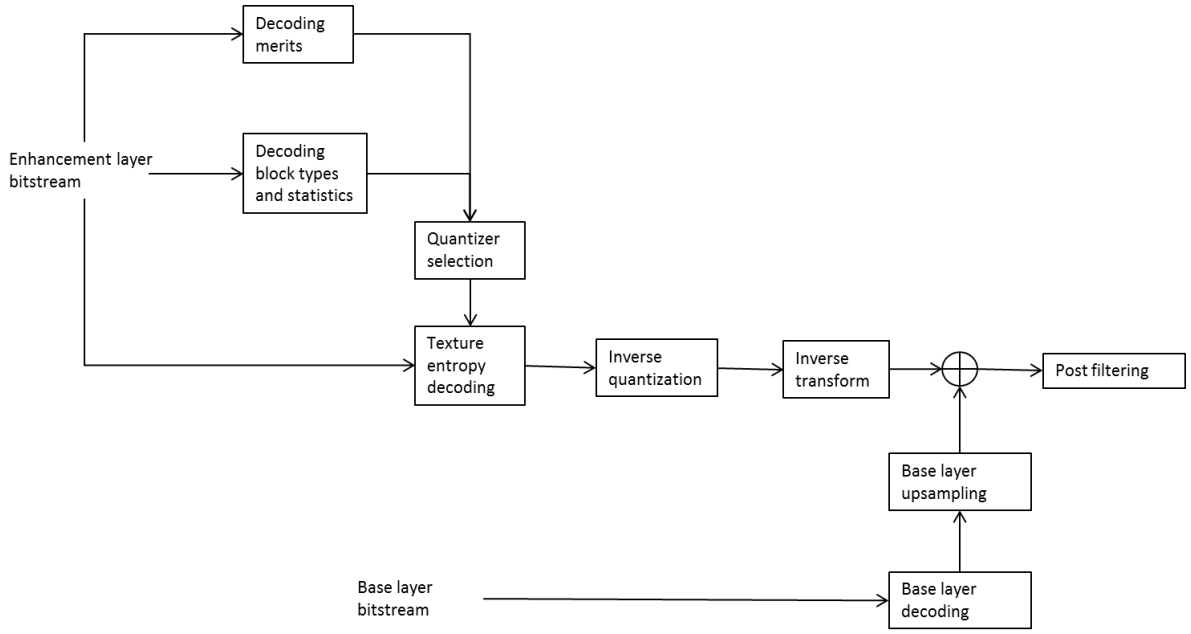

Figure 3: Synoptic of intra-only decoding process

The Intra decoding process is also of low complexity since no Inter-dependency between CUs is required, allowing full spatial random access capabilities. The bit-rate balancing process is achieved before decoding the Transform coefficients and implies limited computation complexity.

2.1.2 Inter coding: coding structure

This section describes the CSC codec technology for the coding of scalable P and B slices.

2.1.2.1 Random Access coding structure

Figure 4 illustrates the random access temporal coding structure adopted in the proposed scalable video codec. The random access property means that several access points are enabled in the compressed video stream, i.e. the decoder can start decoding the sequence at some pictures different from the first picture in the sequence. This takes the form of periodic Intra picture (picture with instantaneous decoding refresh) coding as illustrated by Figure 4. In addition to Intra pictures, the random access coding structure makes use both of forward and backward temporal predictions, through B pictures. The random access configuration also provides temporal scalability features, which takes the form of the hierarchical B pictures organization of Figure 4. The temporal codec structure used in the enhancement layer is identical to the base layer one, which corresponds to the Random Access HEVC testing conditions employed so far.

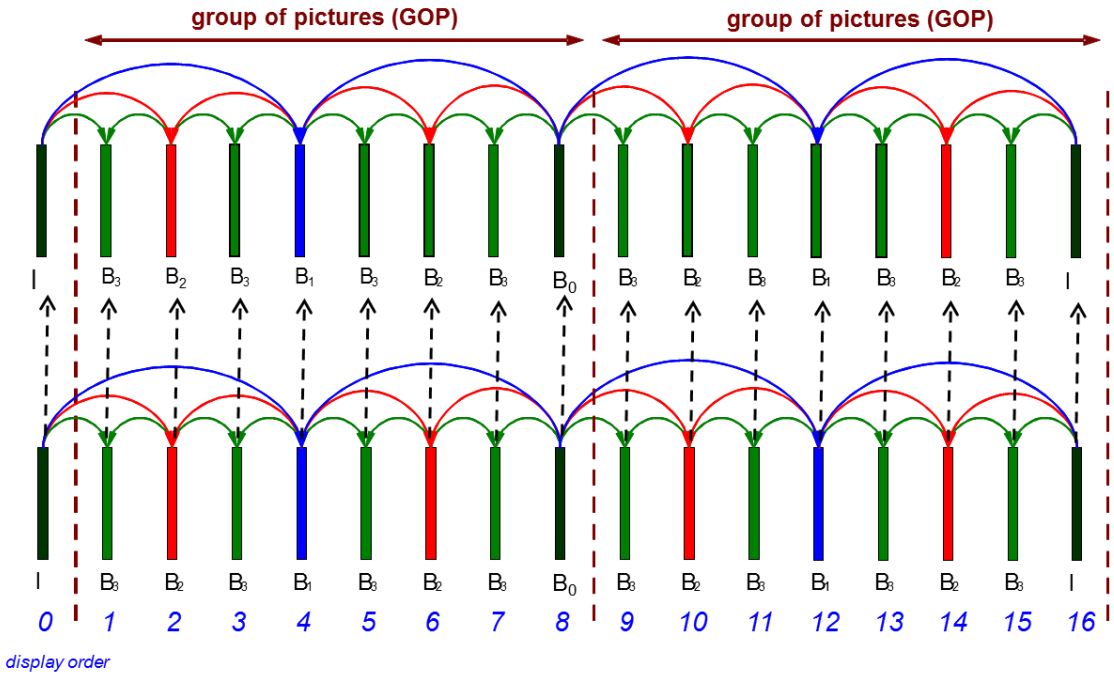

Figure 4: coding structure in Random Access configuration

In the proposed scalable HEVC codec, Intra enhancement picture are coded exactly the same way as in All-Intra configuration. In particular, this involves the base picture up-sampling described in section 2.2 and the texture coding/decoding process described in section 2.3.
With respect to INTER enhancement pictures, the coding and decoding processes are respectively introduced in sections 2.1.3 and 2.1.4, and are fully described in section 2.4.

2.1.3 CSC encoder overview

Figure 5 illustrates a block diagram of the proposed scalable video encoder, illustrated for two spatial or SNR layers. It is made of two stages, respectively dedicated to the coding of the HEVC base layer and the HEVC enhancement layer on top of it. In each stage, closed-loop motion estimation and compensation are performed.
The first stage aims at encoding the HEVC compliant base layer. It encodes the base layer exactly the same way as the HM6.1 encoder, Main profile. Next, the second stage performs the coding of an enhancement layer on top of the base layer. This enhancement layer brings a refinement of the spatial resolution (case of spatial scalability) or of the quality (SNR quality) to the base layer. As shown by Figure 5, the coding scheme of this enhancement layer is similar to the base layer one, except that for each Coding Unit of a current picture being compressed, additional prediction modes can be chosen by the coding mode selection module. These new coding modes correspond to Inter-layer prediction modes. Inter-layer prediction consists in re-using the coded data from a lower layer than current refinement layer, as prediction data for the current Coding Unit. The used lower layer is called the reference layer or base layer for the Inter-layer prediction of current enhancement layer. In case the reference layer contains a picture that temporally coincides with current picture, then it is called the base picture of current picture. The co-located block (at same relative spatial position) of the current Coding Unit that has been coded in the reference layer can be used as a reference to predict the current Coding Unit.
The prediction data from the base picture that can be used in the predictive coding of an enhancement Coding Unit includes the CU prediction information, the motion data (if present) and the texture data (temporal residual or reconstructed base samples).
The **Inter-layer prediction tools** used in the coding of enhancement P and B pictures of the CSC proposal are the following ones.

- **Intra BL** mode consists in predicting a Coding Unit thanks to its co-located area in the up-sampled decoded base picture. This coding mode is detailed in section 2.4.1.1.
- **Base Mode** prediction consists in predicting a Coding Unit from its co-located area in a so-called **Base Mode prediction picture**, constructed both on the encoder and decoder sides. See section 2.4.1.2 for detailed explanation of this prediction mode.
- **Intra Diff** mode consists in a slight modification of HEVC Intra spatial prediction. Here the spatial intra prediction takes place in the differential domain between the enhancement original picture and the up-

sampled, reconstructed base picture. The Intra diff coding mode provided in the CSC codec is coming from [4].

- **Generalized Residual Inter-Layer Prediction (GRILP)** consists in predicting the temporal residual of an INTER Coding Unit, from a temporal residual computed between reconstructed base pictures. See section 2.4.1.3.
- **Inter layer prediction of motion information** attempts to exploit the correlation between the motion vectors coded in the base picture and the motion contained in the topmost layer. It is explained in section 2.4.1.4.

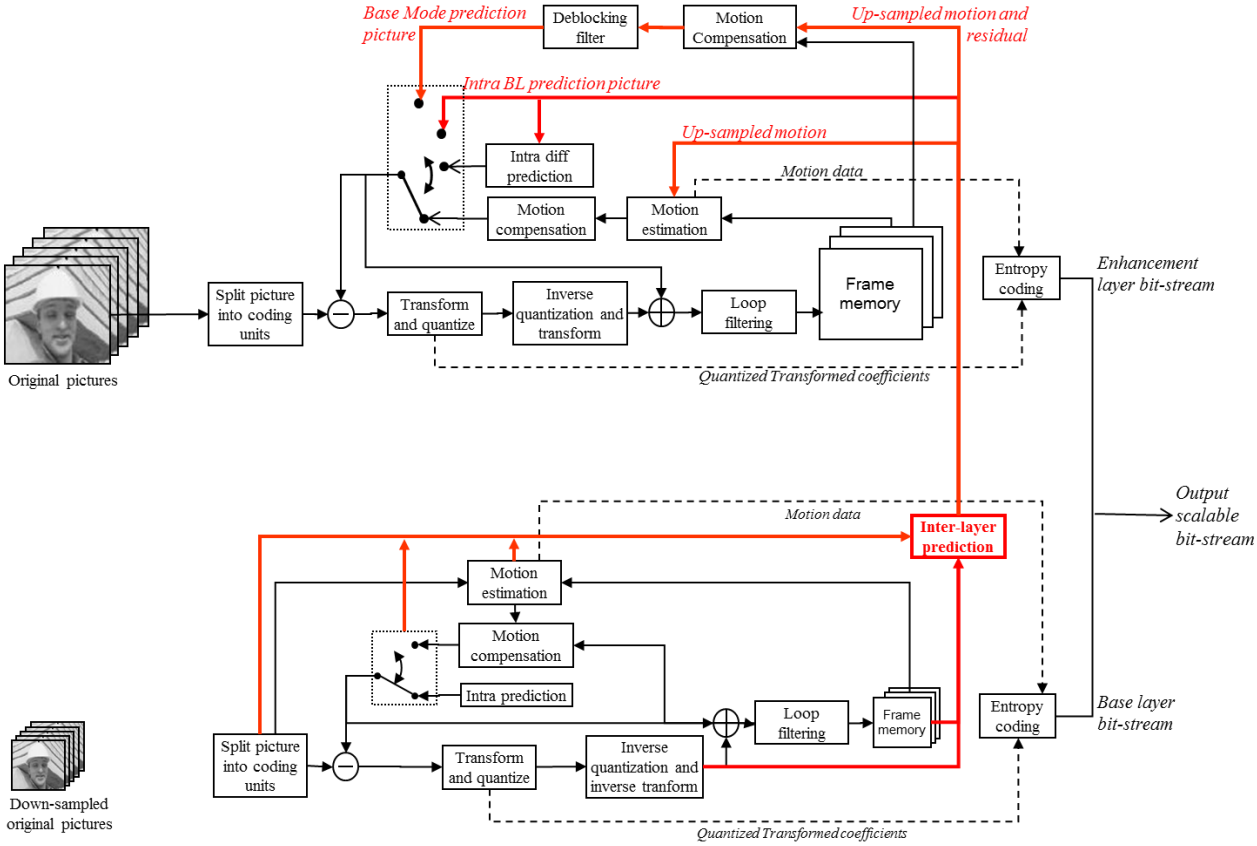

Figure 5: Overview of the proposed scalable video encoder

2.1.4 CSC decoder overview

Figure 6 presents a block diagram of the proposed scalable video decoding framework. The first stage of Figure 6 concerns the base layer decoding process, which is identical to the HM6.1 decoding process.
Next, the second stage performs the decoding of a spatial enhancement layer on top of the base layer decoded by the first stage. This spatial enhancement layer decoding involves the entropy decoding of the second layer, which provides the coding modes, motion information as well as the transformed and quantized residual information.

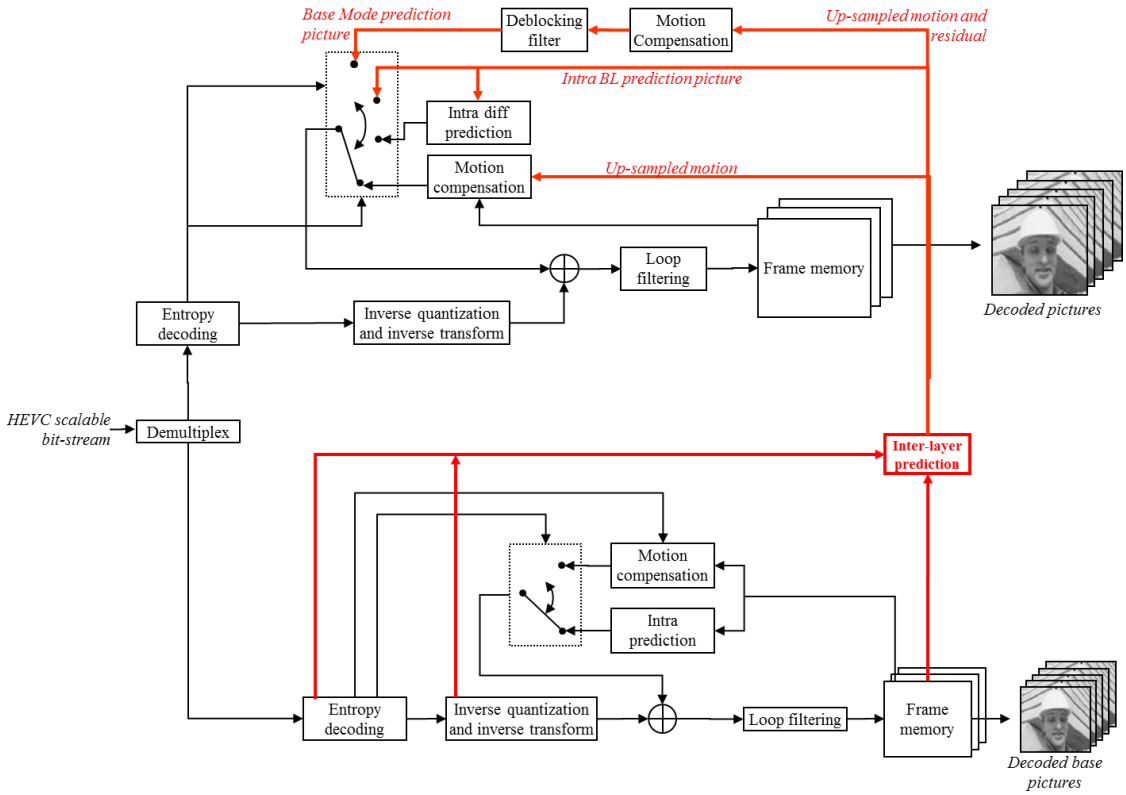

Figure 6: Block diagram of the proposed scalable HEVC decoder

The prediction of each enhancement Coding Unit depends on its coding mode signaled in the bit-stream. The following takes place, according to the CU coding mode.

- In case of an INTER Coding Unit, it is reconstructed according to the HEVC specification. In addition, Inter-layer prediction can be used in two ways for INTER Coding Units. First, its temporal residual data may be predicted from the base layer, through **Generalized Residual Inter-Layer Prediction** described in section 2.4.1.3. Second the motion vectors of PUs contained in current CU are reconstructed in a predictive way, similar to the motion vector prediction in HEVC. This motion vector prediction is enriched in the proposed scalable HEVC extension, so as to enable **Inter-layer motion vector refinement**, as specified in section 2.4.1.4.
- **Intra-BL** prediction type (see section 2.4.1.1): the result of the entropy decoding undergoes inverse quantization and inverse transform, and then is added to the co-located area of current CU in base picture, in its decoded, post-filtered and up-sampled (in case of spatial scalability) version.
- **Base-Mode** prediction (see section 2.4.1.2): the result of the entropy decoding undergoes inverse quantization and inverse transform, and then is added to the co-located area of current CU in the Base Mode prediction picture. The construction of this Base Mode prediction picture is depicted in section 2.4.1.2.
- In case of an Intra Coding Unit, it is reconstructed according to the difference Intra prediction mode, as described in [4]. It is reconstructed through inverse quantization, inverse transform to obtain the residual data in the spatial domain. Intra prediction is performed in the differential domain between the enhancement and the base picture. Then considered CU is reconstructed by adding the predicted CU, the co-located base CU, and the decoded Intra residual.

Note that the Intra BL prediction mode is allowed for every CU in the enhancement picture, regardless the coding mode that was employed in the co-sited Coding Unit(s) of a considered enhancement CU. Therefore, the proposed approach consists in a multiple loop decoding system, i.e. the motion compensated temporal prediction loop is involved in each scalability layer on the decoder side.

2.2 Up-sampling filters

Texture up-sampling process is applied to each reconstructed base picture. The up-sampled base picture is then used in two ways. First, it is used to entirely predict enhancement Intra pictures, as already introduced in section 2.1. Second, it is used to generate the so-called Intra BL prediction picture, which is then used to compute the Intra BL prediction process within INTER enhancement pictures.

The texture up-sampling process involves a 2D separable interpolation filter. It performs vertical and then horizontal interpolation.

The interpolation filter used in the texture up-sampling process consists in a DCT-IF filter, analogous to that used in HEVC motion compensation [1]. 8-tap interpolation filters are used for the Luma component, and 4-tap interpolation filters are used for the two Chroma components.

Phase	Luma Filter coefficients	Chroma Filter Coefficients
0	{ 0, 0, 0, 64, 0, 0, 0, 0}	{ 0, 64, 0, 0}
1/3	{-1, 4, -11, 52, 26, -8, 3, -1}	{-5, 50, 22, -3}
½	{-1, 4, -11, 40, 40, -11, 4, -1}	{-4, 36, 36, -4}
2/3	{-1, 3, -8, 26, 52, -11, 4, -1}	{-3, 22, 50, -5}

Table 1: Phase and filter coefficients used in the texture up-sampling process

Concerning spatial scalability with scaling ratio 2, the phase values used in the interpolation process are 0 and 1/2, as illustrated by Figure 7.

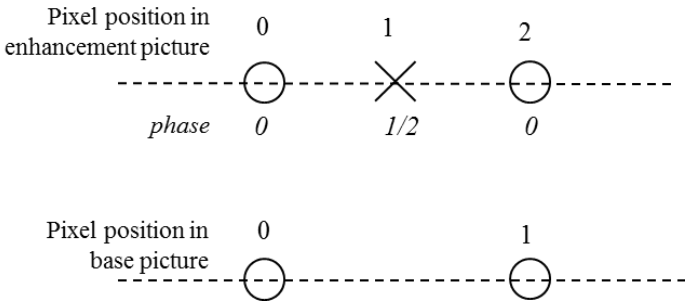

Figure 7: up-sampling process in dyadic spatial scalability mode

Figure 8 illustrates the up-sampling process performed in case of scaling ratio 1.5 between the base layer and the enhancement layer. As can be seen, the phase values involved in this case are 0, 2/3 and 1/3.

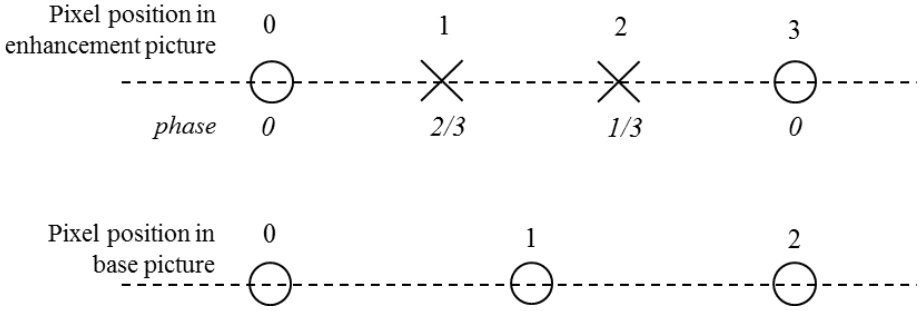

Figure 8: Up-sampling process in spatial scalability with ratio 1.5

2.3 Intra texture coding/decoding process of CSC

This section describes in details the CSC technology for enhanced Intra slices. This technology is used for all pictures of the intra-only CfP configuration, and to the Intra pictures of the Random Access of the CfP configuration. An overview of the technology is presented above, in section 2.1.1. The following sub-sections explain the different elementary steps involved in the full encoding and decoding processes.

2.3.1 Block (TU) partitioning

This section describes the segmentation of the intra enhancement residual image in transform units of different sizes and different types. The standard HEVC transforms are then applied to each TU depending only of their size.

2.3.1.1 Block type: Block size and Block label

The concept of block size has been generalized to what will be called **block types**.
A block type is the given of

- a block size,
- a block label.

It is well known that, most of time, a region of a frame with a lot of activity is better compressed by using small blocks; on the contrary less active (somewhat flat) regions are better compressed by using large blocks.
In HEVC, squared transform unit sizes are either 32x32, 16x16, 8x8 or 4x4. We used a similar approach but only with three squared sizes **32x32, 16x16 and 8x8 for the luminance component** Y, and the corresponding smaller sizes **16x16, 8x8 and 4x4 for the two chrominance components** U and V as we only consider 4:2:0 videos.
One also notices that there may be a big disparity of activity (or energy) between blocks with the same size. It turns out that a segmentation of a frame by using only block size is not fine enough to obtain an optimal performance of classification of parts of the frame. That is why a **label** has been added to the block size in order to distinguish **various levels and/or characteristics of a block activity**.
In the framework of the scalable codec, this activity is the activity of the residual frame which is the difference between the original frame and the decoded base layer. It is called **residual activity.**

More precisely, we use the following types for the coding of the residual enhancement layer.

- block type 32x32 label skip
- block type 32x32 label 1
- …
- block type 32x32 label N_{32}

- block type 16x16 label skip
- block type 16x16 label 1
- …
- block type 16x16 label N_{16}

- block type 8x8 label skip
- block type 8x8 label 1
- …
- block type 8x8 label N_8

As one may expect, a block type with a label “skip” does not encode the texture (= all coded residual samples, or equivalently all DCT coefficients, are set to zero).
There are N_{32}+1 block types of size 32x32, are N_{16}+1 block types of size 16x16 and are N_8+1 block types of size 8x8. The optimal choice of the parameters N_{32}, N_{16}, N_8 depends on the residual frame content and, as a general rule, high quality coding requires more block types than low quality coding.

2.3.1.2 Block types and components Y, U and V

The same segmentation into block types is shared for the three components Y, U and V. Practically, a block type **NxN label L for a YUV block** (a YUV block is defined as a block Y with its associated blocks U and V) underlies the following inference for each component:

- block type **NxN label L_Y** for the block NxN in Y,
- block type **N/2xN/2 label L_U** for the block N/2xN/2 in U,
- block type **N/2xN/ label L_V** for the block N/2xN/2 in V.

A subscript of the component name has been added to the label because, as we will see later, the coding depends also on the component. For instance, the coding of N/2xN/2 label L_Y is not the same as the coding of N/2xN/2 label L_U since associated quantizers differ. Similarly, the coding of N/2xN/2 label L_U differs from the coding of N/2xN/2 label L_V.

2.3.1.3 Block types and quad-tree coding

The segmentation of a frame into block types is performed through the syntax of a quad-tree. In HEVC, the level NxN of the quad-tree takes only two values

- split into four smaller blocks N/2xN/2,
- or no split.

The coding of block types uses a generalized quad-tree with more than two values per level

- level 32x32: 0=skip, 1=label 1, …, N_{32}=label N_{32}, N_{32}+1 = split into 16x16 blocks,
- level 16x16: 0=skip, 1=label 1, …, N_{16}=label N_{16}, N_{16}+1 = split into 8x8 blocks,
- level 8x8: 0=skip, 1=label 1, …, N_8=label N_8.

The generalized quad-tree representation is compressed by using an arithmetic entropy coder. For a label L of the quad-tree, the probability

$$\mathbf{P(L|s_B)}$$

is computed, where **s_B is the state of the co-located base layer block**. These probabilities must be sent to the decoder to ensure decodability of the quad-tree by a context-free arithmetic coder.
Typically, we use the pixel morphological energy (only taken from the luminance component) of the base block as a guess for defining the state s_B. The range of possible energies is divided in several bands and the value of a state is just the number of the band the energy belongs to.

2.3.1.4 Initialization of the segmentation into block types

The initial segmentation of the residual frame into various block types is performed following the analysis of the **residual morphological activity**. This choice is not restrictive and other methods of activity evaluation (for instance Laplace filter) provide similar performance. The basics of mathematical morphology can be found in [7] or [8].

The choice of the block type is performed by a top-down algorithm in term of block size. One starts from the coarsest 32x32 block division and the **L_2 integral I of the morphological gradient** on each 32x32 block is computed. If this integral is higher than a fixed threshold, the block is divided into four smaller 16x16 blocks. This process is applied on all obtained 16x16 blocks to decide whether or not they are divided into 8x8 blocks.
Once the segmentation into block sizes is decided, one has to determine the label for each block to find out the initial block type segmentation. The label decision is based not only on the magnitude of the integral I but also on the ratio vertical vs. horizontal activity defined as follows:

$$I_h/I_v$$

where I_h is the L_2 integral I of the horizontal morphological gradient and I_v is the L_2 integral I of the vertical morphological gradient.
The initial segmentation determined by arbitrary thresholds on I and I_h/I_v may not be optimal. We show later how to improve it by a rate-distortion optimization.

2.3.2 Modeling of the residual frame per block type

For a given block type, the corresponding DCT transform is applied and then we fit a model for the channel of the DCT coefficients of this block type.

2.3.2.1 Generalized Gaussian Distributions

It is common to model the DCT residual of video data by a **Generalized Gaussian Distribution (GGD)**. Naturally, this distribution has zero mean, such that one may claim the approximation

$$\mathrm{DCT}\ X \approx GGD(\alpha, \beta)$$

where the GGD follows the two-parameter distribution

$$GGD(\alpha, \beta, x) := \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp(-|x/\alpha|^{\beta})$$

The **standard deviation** σ of the distribution is proportional to the parameter α with a corrective factor depending on β.

$$\sigma^2 = \alpha^2\Gamma(3/\beta)/\Gamma(1/\beta)$$

The parameter β **is called the exponent of the distribution**.

2.3.2.2 Fitting of Generalized Gaussian Distributions

The moment of order k of the absolute value of a GGD is defined as follows

$$M_k^{\alpha,\beta} := E\left(|GGD(\alpha, \beta)|^k\right) \qquad (k \in R_+).$$

By simple integral calculation, one finds out an analytical formula for the moments.

$$M_k^{\alpha,\beta} = \int_{-\infty}^{\infty} |x|^k GGD(\alpha, \beta, x)\, \mathrm{d}x = \frac{\alpha^k\Gamma((1+k)/\beta)}{\Gamma(1/\beta)}$$

It is also observed that well-chosen ratios of moments do not depend on the parameter α ; for instance we get

$$\frac{M_2}{M_1^2} = \frac{\Gamma(1/\beta)\Gamma(3/\beta)}{\Gamma(2/\beta)^2}$$

So, for modeling a random variable by a GGD, it is easy to estimate the value of the parameter β by computing the above ratio of the two first and second moments and then inversing the above function of β. Practically, instead of computing costly Gamma functions, this inverse function may be tabulated.
Finally, the parameter α is estimated thanks to the second moment by the equality

$$M_2 = \sigma^2 = \alpha^2\Gamma(3/\beta)/\Gamma(1/\beta).$$

This method is very simple and robust for determining a GGD model of a random variable.

2.3.2.3 Modeling of transformed blocks

We observed experimentally that the DCT coefficients of the spatial residual are well modeled by GGD's. Of course, they cannot be all modeled by the same parameters and, practically, the parameters depend on

- the video content; this means that the parameters must be computed every frame or every *n* frames for instance
- the index of the DCT coefficient; each DCT coefficient has its own behavior
- the block type.

For instance, for a block type of size 8x8, the parameters of 64 channels for the luminance component Y and 16 channels for each chrominance component U and V are derived. It has been observed that U and V have dramatically different source contents in many videos, so it is better to compute separate models for U and V.
This may seem to be a lot of parametric data to be coded into the video stream, but the experience proves that this is quite negligible compared to the volume of data needed to encode the residues of High Definition or bigger videos. It is obvious that such a technique could not be used for very small videos because the parametric data would be too costly.
We will also show later how to reduce the cost of statistics by avoiding re-computing for each frame.

2.3.3 The rate-distortion coding problem for one block type

2.3.3.1 Pixel distortion and control of DCT coefficients

Usually, the objective quality of a video is measured by the so-called PSNR which is a measure of the L_2-norm of the error of encoding in the pixel domain. However, most of video codecs compress the data in the DCT-transformed domain in which the energy of the signal is much better compacted. Here, we show the direct link between the pixel distortion after coding and the error on DCT coefficients.

Let us consider a residual block and let ψ_n be its IDCT pixel base in the pixel domain as shown on Figure 9. If one uses the so-called IDCT III for the inverse transform, this base is orthonormal

$$\|\psi_n\| = 1.$$

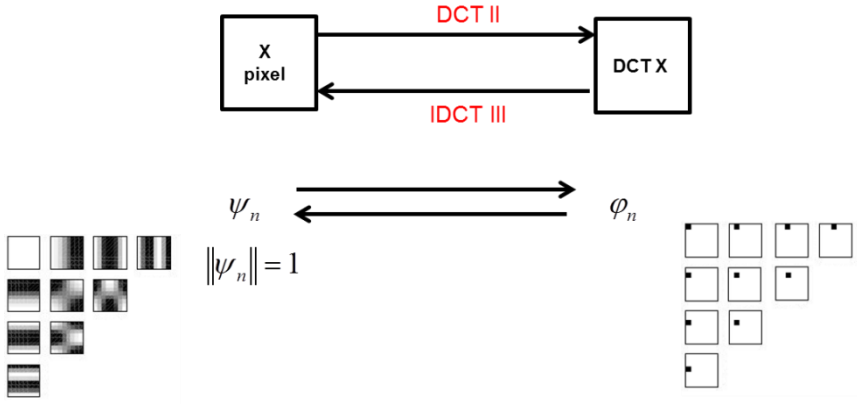

Figure 9: Pixel and DCT bases of a block

On the other hand, in the DCT domain, the unity elements (=zero everywhere except one at one location) form a base φ_n which is orthogonal. The DCT transform of the pixel block X is written as follows

$$\text{DCT } X = \sum_n d^n \varphi_n$$

where d^n is the value of the n-th DCT coefficient. A simple base change leads to the expression of the pixel block as a function of the DCT coefficient values.

$$X = \text{IDCT}(\text{DCT } X) = \text{IDCT}\sum_n d^n \varphi_n = \sum_n d^n \text{IDCT}(\varphi_n) = \sum_n d^n \psi_n$$

Let us denote d_Q^n the value of the de-quantized coefficient d^n after decoding. By linearity, one sees that the pixel error block is given by

$$\varepsilon_X = \sum_n (d^n - d_Q^n)\psi_n$$

The mean L_2-norm error on all blocks (pixel distortion), is computed as

$$E\left(\|\varepsilon_X\|_2^2\right) = E\left(\sum_n \left|d^n - d_Q^n\right|^2\right) = \sum_n E\left(\left|d^n - d_Q^n\right|^2\right) =: \sum_n D_n^2$$

where D_n^2 is the mean quadratic error of quantization on the n-th DCT coefficient. So, the control of the video quality is equivalent to the control of the sum of the quadratic errors on DCT coefficients. In particular, the individual control of each one of the DCT coefficient is a priori a sub-optimal control.

2.3.3.2 Rate minimization under distortion constraint

Let us suppose that the PSNR target corresponds to a given L_2 distortion value D_t^2. The minimization of the rate under this pixel distortion constraint can be stated as

$$\text{minimize } R = \sum_n R_n(D_n) \quad \text{s.t.} \quad \sum_n D_n^2 = D_t^2 \quad \textbf{(A)}$$

where R is the total rate made of the sum of individual rates R_n for each DCT coefficient. In case the quantization is made independently for each DCT coefficient, the rate R_n depends only on the distortion D_n of the associated n-th DCT coefficient.

The above minimization problem is fulfilled only by optimal quantizers which are solution of the problem

$$\text{minimize } R_n(D_n) \quad \text{s.t.} \quad E\left(\left|d^n - d_Q^n\right|^2\right) = D_n^2 \quad \textbf{(B)}$$

This statement is simply proven by the fact that if a quantizer is not optimal, then another quantizer with lower rate but the same distortion can be constructed (or obtained). So, if this other quantizer is used, the total rate R is diminished without changing the total distortion $\sum_n D_n^2$; this is in contradiction with a minimal solution of the first problem **(A)**. So, we have just shown that the rate-distortion minimization problem **(A)** can be split into two consecutive sub-problems without changing the optimality of the solution.

- first, determining optimal quantizers and their associated rate-distortion curves $R_n(D_n)$ following the problem **(B)** for GGD channels.
- second, we solve the problem **(A)** changed into the problem **(A_{opt})** where quantizers are optimal quantizers

$$\text{minimize } R = \sum_n R_n(D_n) \quad \text{s.t.} \quad \sum_n D_n^2 = D_t^2 \text{ and } R_n(D_n) \text{ is optimal} \quad \textbf{(A_{opt})}$$

The resolution of the problem (A_{opt}) is described hereafter in the following sections.

2.3.4 Off-line determination of optimal quantizers

This section deals with the derivation of optimal quantizers. We consider only 1D quantizers for a given DCT coefficient. This means that one quantizer will be used per DCT coefficient for the quantization process.

2.3.4.1 Voronoi cell based quantizers

A quantizer is made of M Voronoi cells which, in 1D, are intervals $[t_m, t_{m+1}]$ called quantum Q_m , and each cell has a centroid d_m , as shown on Figure 10. The intervals are used for quantization while centroids are used for de-quantization: a value falling in an interval is coded by the index m of this interval and is de-quantized into the centroid value d_m .

Let us suppose that the probability distribution $P(x)$ of the coefficient is given; for instance a GGD as seen on the preceding Figure 10. The distortion D_m of the quantization, on the quantum Q_m , is the mean error $E(d(x;d_m))$ for a given distortion function d . Usually, this function is a L^2 -distance on the real line. So the distortion on one quantum is given by

$$D_m^2 = \int_{Q_m} |x - d_m|^2 P(x) \mathrm{d}x$$

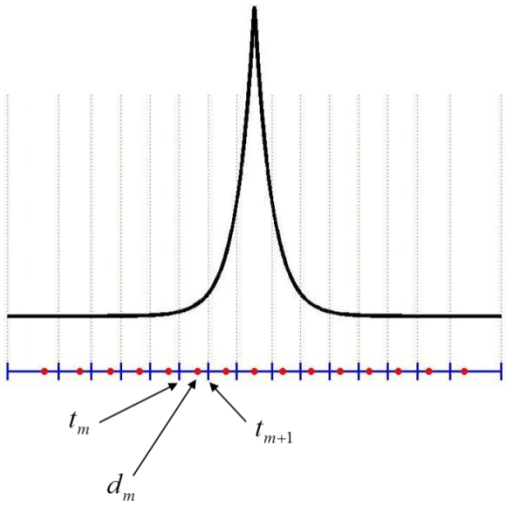

Figure 10: A 1D quantizer made of Voronoi cells

and the total distortion is

$$D^2 = \sum_m D_m^2 = \sum_m \int_{Q_m} |x - d_m|^2 P(x) \mathrm{d}x$$

The total rate R is the cost of encoding by an entropic code close to the entropy. At best, this rate reaches the entropy of the quantized data

$$H = -\sum_{m=1}^{M} P_m \log_2 P_m$$

where P_m is the probability of x to be in the quantum Q_m and is simply the following integral

$$\boxed{P_m = \int_{Q_m} P(x) \mathrm{d}x}.$$

Practically, the rate-distortion problem may be stated in two different ways.

- rate control: for a given rate R, find the quantization which minimizes the distortion D
- distortion control: for a given distortion D, find the quantization which minimizes the rate R

This problem is hard to solve because it is a non-linear minimization problem.

2.3.4.2 Lagrangian formulation (B_λ) of the problem (B)

Instead of the previous complex formulation, we use the so-called Lagrange formulation of the problem. For a given parameter $\lambda > 0$, we determine the quantization in order to minimize the cost function $D + \lambda R$. This way, we get an optimal rate-distortion couple (D_λ, R_λ). In case of a rate control for a given target distortion D_t, the optimal parameter $\lambda > 0$ is determined by

$$\lambda_{D_t} := \arg\min_{\lambda,\, D_\lambda \le D_t} R_\lambda$$

and the associated minimum rate is

$$R_{D_t} := R_{\lambda_{D_t}}$$

So, it is possible to plot a rate distortion curve $D_t \mapsto R_{D_t}$ which may be computed off-line as well as the associated quantifications. So, we restated problem (B) into a continuum of problems (B_λ)

$$\text{minimize } D_n^2 + \lambda R_n(D_n) \quad \text{s.t.} \quad E\left(|x - d_m|^2\right) = D_n^2 \qquad (B_\lambda)$$

One understands that the cost can be changed from $D + \lambda R$ to $D^2 + \lambda R$ without loss of generality thanks to the extra parameter λ.

2.3.4.3 The Chou-Lookabaugh-Gray algorithm

Still remains the minimization of the linear problem (B_λ) for $D^2 + \lambda R$ to determine the quantization. The well-known Chou-Lookabaugh-Gray algorithm is a good practical way to perform it. It can deal with any distortion distance d, but here we present a simplified algorithm for the L^2-distance. This is an iterative process from any given starting guessed quantization.

The position of the centroids d_m is easy to determine because they must minimize the distortion D_m^2 inside a quantum, in particular that the condition $\partial_{d_m} D_m^2 = 0$ must be verified. We easily deduce

$$\boxed{d_m = \int_{Q_m} xP(x)\mathrm{d}x / P_m}.$$

Now, let us consider the cost function

$$C = D^2 + \lambda R$$

which must be minimized. In particular its derivatives must vanish for an optimal solution.

$$0 = \partial_{t_{m+1}} C = \partial_{t_{m+1}} \left[D_m^2 - \lambda P_m \ln P_m + D_{m+1}^2 - \lambda P_{m+1} \ln P_{m+1} \right]$$

Let us set $\overline{P} = P(t_{m+1})$ the value of the probability distribution at the point t_{m+1}. From simple variational considerations, see Figure 11, we get

$$\partial_{t_{m+1}} P_m = \overline{P} \qquad \text{and} \qquad \partial_{t_{m+1}} P_{m+1} = -\overline{P}\ .$$

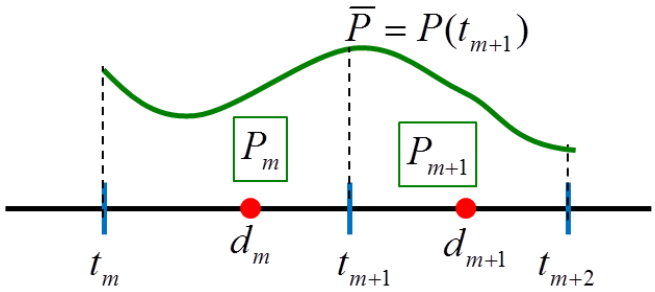

Figure 11: Effect of the variation of the position of point t_{m+1}

Then, a bit of calculation leads to

$$\begin{aligned}
\partial_{t_{m+1}} D_m^2 &= \partial_{t_{m+1}} \int_{t_m}^{t_{m+1}} |x - d_m|^2 P(x)\mathrm{d}x \\
&= \overline{P} |t_{m+1} - d_m|^2 + \int_{t_m}^{t_{m+1}} \partial_{t_{m+1}} |x - d_m|^2 P(x)\mathrm{d}x \\
&= \overline{P} |t_{m+1} - d_m|^2 - 2\partial_{t_{m+1}} d_m \int_{t_m}^{t_{m+1}} (x - d_m) P(x)\mathrm{d}x \\
&= \overline{P} |t_{m+1} - d_m|^2
\end{aligned}$$

as well as

$$\partial_{t_{m+1}} D_{m+1}^2 = -\overline{P} |t_{m+1} - d_{m+1}|^2 .$$

Now, the derivative of the cost is explicitly calculated

$$0 = \overline{P} |t_{m+1} - d_m|^2 - \lambda \overline{P} \ln P_m - \lambda P_m \frac{\overline{P}}{P_m} - \overline{P} |t_{m+1} - d_{m+1}|^2 + \lambda \overline{P} \ln P_{m+1} + \lambda P_m \frac{\overline{P}}{P_m},$$

This leads to a useful relation between the quantum boundaries and the centroids

$$\boxed{t_{m+1} = \frac{d_m + d_{m+1}}{2} - \lambda \frac{\ln P_{m+1} - \ln P_m}{2(d_{m+1} - d_m)}}.$$

This formula is already well-known, but a simple demonstration has been given for the sake of completeness.

The algorithm by Chou-Lookabaugh-Gray is an iterative process as described below.

1. Start with arbitrary quanta t_m
2. Compute the probabilities P_m by the formula

   $$P_m = \int_{Q_m} P(x)\mathrm{d}x$$

3. Compute the centroids d_m by the formula

   $$d_m = \int_{Q_m} xP(x)\mathrm{d}x / P_m$$

4. Compute the new quanta t_m by the formula

   $$t_{m+1} = \frac{d_m + d_{m+1}}{2} - \lambda \frac{\ln P_{m+1} - \ln P_m}{2(d_{m+1} - d_m)}$$

5. Compute the cost $C = D^2 + \lambda R$ by the formula

   $$C = \sum_m D_m^2 - \lambda P_m \ln P_m$$

6. Loop to 2. until convergence of the cost C

So, for a given number M of quanta, each problem (B_λ) is solved by this algorithm.

2.3.4.4 Normalization

One easily understands that the parameter α (or equivalently the standard deviation σ of the GGD) does not play a role on the determination of optimal quantizers because it is an homothetic parameter which can be moved out from the distortion parameter D_n^2.

So, only optimal quantizers with unity standard deviation $\sigma = 1$ must be determined and the GGD will be normalized before quantization, and then be de-normalized after de-quantization. Of course, this is possible because the parameter σ of the GGD models have also been sent to the decoder through the video bit-stream.

2.3.4.5 The upper envelope of optimal quantizers. Tabulation

For a given parameter β, the problems (B_λ) are solved for many values of the Lagrange parameter λ and for various odd (by symmetry) values of the number M of quanta. Thus, a rate-distortion diagram for the optimal quantizers can be built, as shown on Figure 12.

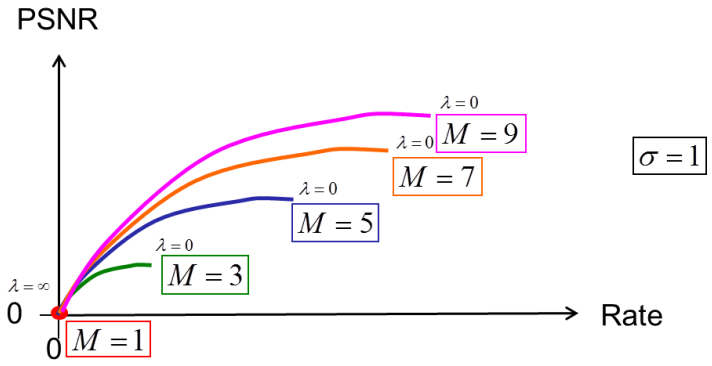

Figure 12: The rate-distortion curves associated to optimal quantizers for the problems (B_λ)

It turns out that, for a given distortion, there is an optimal number of needed quanta for the quantization. In brief, one may say that optimal quantizers of the general problem (B) are those associated to a point of the upper envelope of the rate-distortion curves making this diagram. This upper envelope is illustrated on Figure 13.

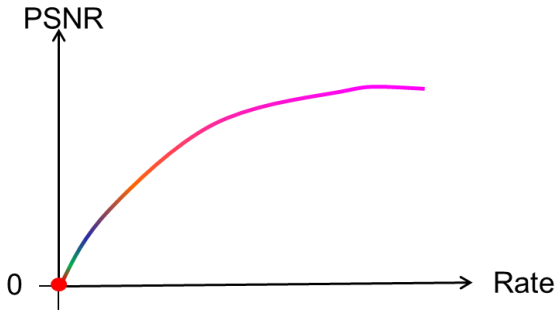

Figure 13: The upper envelope of the rate-distortion curves associated to optimal quantizers for the whole set of parametric problems (B_λ)

Practically, optimal quantizers, for all parameters β and all target distortions D_n^2, cannot be computed. It is experimentally observed that the GGD modeling provides β's almost always between 0.5 and 2, and that only a few discrete values are needed for the precision of encoding. In our codec, β is tabulated every 0.2 in the interval [0.6, 2.0]; thus, there are 8 values for β, encoded on 3 bits through a look-up table.

In the continuous domain, for each value of β, a rate-distortion curve is obtained as shown on Figure 14.

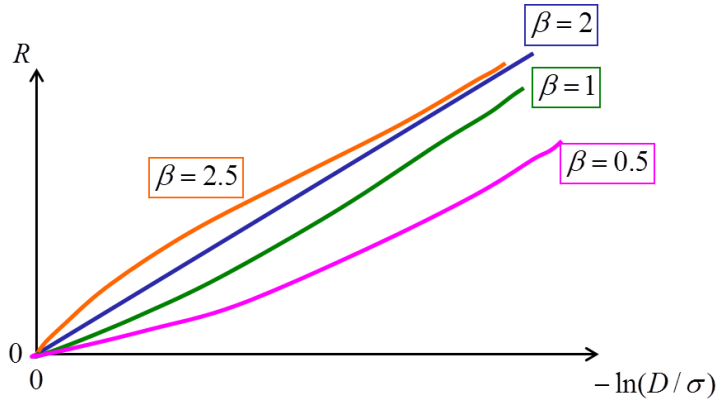

Figure 14: The rate-distortion curves of optimal quantizers for GGD channels

About two hundreds (number slightly depending on β) of quantizers are stored for each of the 8 tabulated β. Quantizers are computed up to a maximum rate of about 6 bits per DCT channel.

2.3.5 Balanced encoding of DCT coefficients of a block type

2.3.5.1 Rate-distortion model of optimal quantizers

From the rate-distortion curves on Figure 14, one notices that, as expected, the rate decreases monotonously as a function of the distortion induced by the quantizer. So, the following equation can be written

$$R_n = f_n(-\ln(D_n / \sigma_n))$$

where σ_n is the normalization factor of the DCT coefficient, i.e. the GGD model associated to the DCT coefficient has σ_n for standard deviation. In particular, no encoding (equivalently zero rate) leads to a quadratic distortion of value σ_n^2 and we deduce the equality at the origin

$$0 = f_n(0).$$

The monotonicity gives

$$f_n' \geq 0$$

and finally, one observes that the curves are convex for parameters β lower than two.

$$\boxed{\beta \leq 2 \quad \Rightarrow \quad f_n'' \geq 0}$$

2.3.5.2 Merit of encoding

Now, let us consider the merit of encoding a DCT coefficient. More encoding basically means

- more rate R_n, in other words "the price to pay";
- less distortion D_n^2, in other words "the gain".

When one wants to dedicate a bit more rate to the encoding of the video, one must determine on which DCT coefficient this "extra rate" is the most efficient. The **merit of encoding** is the ratio of benefit of distortion on cost of encoding

$$M := \left| \frac{\Delta D^2}{\Delta R} \right|.$$

Of course, this is just a way of revising the "lambda factor" in the framework of our codec, but it may be useful to use a new terminology to avoid collisions with more standard terminology later.
Let us suppose that we reduce the distortion by an amount of ε, then a first order development of distortion and rates gives

$$(D-\varepsilon)^2 = D^2 - 2\varepsilon D + o(\varepsilon)$$

and

$$\begin{aligned} R(D-\varepsilon) &= f_n(-\ln((D-\varepsilon)/\sigma)) \\ &= f(-\ln(D/\sigma) - \ln(1-\varepsilon/D)) \\ &= f(-\ln(D/\sigma) + \varepsilon/D + o(\varepsilon)) \\ &= f(-\ln(D/\sigma)) + \varepsilon f'(-\ln(D/\sigma))/D \end{aligned}$$

Then, the ratio of the first variations provides an explicit formula for the merit of encoding.

$$M_n(D_n) = \frac{2D_n^2}{f'(-\ln(D_n/\sigma_n))}$$

For β's lower than two, the convexity of the rate distortion curves teaches us that the merit is an increasing function of the distortion.

$$\beta \leq 2 \quad \Rightarrow \quad M_n \uparrow D_n$$

The **initial merit** is, by definition, the merit of encoding at zero rate, i.e. before any encoding.

$$M_n^0 := M_n(\sigma_n) = \frac{2\sigma_n^2}{f'(0)}$$

2.3.5.3 The theorem of encoding at equal merit

A mathematical formula for the optimal amount of encoding of DCT coefficient, such that a given distortion D_t^2 is reached, can be determined by using the so-called KKT (Karash, Khun, Tucker) method. We skip all the details of lengthy calculations and claim that one gets

$$D_n^2 = \left(D_t^2 - \sum_{I^+} \sigma_n^2 \right) f_n'(-\ln(D_n/\sigma_n)) / \sum_{m \in I^0} f_m'(-\ln(D_m/\sigma_m)) \quad (n \in I^0) \ .$$

where I^0 is the set of coded DCT coefficients and D_t^2 is the distortion target. This equation is implicit in the D_n^2's. This equation is rearranged to obtain

$$M_n = 2\left(D_t^2 - \sum_{I^+} \sigma_n^2 \right) / \sum_{m \in I^0} f_m'$$

From this, we deduce the theorem encoding at equal merits.

The theorem of encoding at equal merits
All encoded DCT coefficients of a block type have the same merit after encoding. The common merit is called ***the merit of the block type****.*

Furthermore, all non-coded coefficients have a merit bigger than the merit of the block type.

Proof: The formula from the KKT algorithm is a proof itself, but we provide another heuristic proof here.

Let's suppose that there are two encoded DCT coefficients with two different merits $M_1 < M_2$. Now, let's take an infinitesimal amount of rate from coefficient 1 to put this rate on coefficient 2. This is possible because coefficient 1 is encoded and this does not change the total rate.

Because $M_1<M_2$, the distortion gain on coefficient 2 is strictly bigger than the distortion loss on coefficient 1. So we get a better distortion with same rate. We just have found a better R(D) point; this would be in contradiction with optimality.

Conclusion: if the two coefficients 1 and 2 are encoded and if $M_1 < M_2$, then the solution is not optimal. This completes the proof.

Note that the theorem of equal merits is similar to the Pareto's condition in the field of source coding [6].

Practically, the model of DCT coefficients always provide $\beta \leq 2$ and the merits are increasing functions of the distortion. As a consequence, as shown on Figure 15, for a given target block merit M_{block}, for each DCT coefficient,

- either the initial merit of the coefficient is lower than the target block merit and the coefficient is not encoded
- or there is a unique distortion D_n^2 such that $M_n(D_n) = M_{block}$.

This unique solution is easily found by dichotomy on the distortion D_n^2.

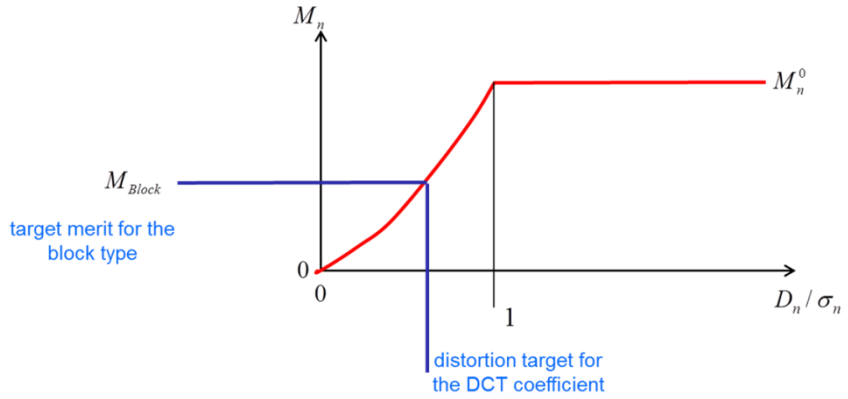

Figure 15: How to find distortions from the target merit for the block type

2.3.6 Balanced encoding between block types

Due to their various sizes, all block types cannot have the same block merit in an optimal coding. Here we show that there is a very simple relation between block merits and the newly introduced frame merit.

2.3.6.1 A few definitions

Let us start from a few straightforward definitions.

- **DCT index number** n
- **block type** $\bullet$
- **distortion of a DCT coefficient** $D^2_{n,\bullet}$
- **rate of a DCT coefficient** $R_{n,\bullet}$
- **pixel distortion of a block type** $D^2_{p,\bullet}$
- **rate of a block type** $R_\bullet$
- **merit of encoding of a block type** $M_\bullet$
- **density of a block type in the frame** $\rho_\bullet$
- **number of blocks per unit of surface** $N_\bullet$
- **rate per unit of surface** $R_{S,\bullet}$
- **frame rate** R_F
- **frame distortion** D_F^2

The correspondence between pixel distortion and DCT coefficient distortion becomes simply

$$D^2_{p,\bullet} = \sum_n D^2_{n,\bullet} ,$$

and the mean rate of encoding for a block type is provided by

$$R_\bullet = \sum_{\text{coded}\, n} R_{n,\bullet}$$

The theorem of equal merits states that optimal encoding must fulfill the following relation.

$$\forall \text{coded } n \quad M_\bullet = M_{n,\bullet}$$

Now, let us care about the effect of the block sizes. First, because the entire frame is covered by blocks, the sum of densities must be unity.

$$1 = \sum_\bullet \rho_\bullet$$

The rate per unit of surface of a block type is the rate of one block multiplied by the number of blocks per unit of surface.

$$R_{S,\bullet} = N_\bullet R_\bullet$$

Then, the mean rate per unit of surface for the whole frame is the sum of surface rates per block type weighted by the densities.

$$R_F = \sum_\bullet \rho_\bullet R_{S,\bullet}$$

Finally, the mean frame distortion is the distortion per block type weighted by the densities.

$$D_F^2 = \sum_\bullet \rho_\bullet D_{p,\bullet}^2$$

2.3.6.2 Merit of the frame and merit of the block type

The balance of encoding between block types can be stated as follows. The merit of encoding per pixel, also called the **merit of the frame** M_F, must be the same for all block types, i.e.

$$\forall \bullet \quad M_F = \frac{\Delta D_{p,\bullet}^2}{\Delta R_{S,\bullet}}.$$

Clearly, just considering the merit of DCT coefficients does not take the surface of the blocks into account. From the equal merit property, we get

$$\forall \text{coded } n \quad M_\bullet \Delta R_{n,\bullet} = \Delta D_{n,\bullet}^2.$$

The variation of rate and distortion for a block type is provided by the variations of coded coefficients as follows.

$$\Delta R_\bullet = \sum_{\text{coded } n} \Delta R_{n,\bullet}$$

$$\Delta D_{p,\bullet}^2 = \sum_{\text{coded } n} \Delta D_{n,\bullet}^2$$

So, we deduce a relation between the variation of pixel distortion and the variation of the rate.

$$\Delta D_{p,\bullet}^2 = M_\bullet \Delta R_\bullet$$

Now, using this equality in the definition of pixel merit, we obtain

$$\forall \bullet \quad M_F = \frac{\Delta D_{p,\bullet}^2}{\Delta R_{S,\bullet}} = \frac{M_\bullet \Delta R_\bullet}{N_\bullet \Delta R_\bullet} = \frac{M_\bullet}{N_\bullet}.$$

This proves the very simple relation between the frame merit and the block merit.

$$\boxed{M_\bullet = N_\bullet M_F}$$

2.3.7 Balanced encoding between frames and color components

One may consider that optimal encoding of a video is obtained by encoding all frames with a constant frame merit. This is not always true; for instance, the quality of a frame is commonly measured by using the PSNR (logarithm of the quadratic samples distortion). Moreover actual measured PSNR is often the PSNR average taken over all frames. So, because the quality of the video is not the mean quadratic pixel distortion of all frames but the mean of logarithms, we must balance the merits between frames to optimize the results for this way of evaluating the quality.

2.3.7.1 Case of the luminance component Y only

Let us write the PSNR as the logarithm of the distortion

$$\mathrm{PSNR} = \ln(D_F^2) .$$

We dropped the constant because they do not change the final result as the reader can easily verify.

A variation of encoding induces a variation of distortion which induces a variation of PSNR as follows,

$$\ln(D_F^2 + \Delta D_F^2) \approx \ln(D_F^2) + \Delta D_F^2 / D_F^2 ,$$

such that we get

$$\Delta\mathrm{PSNR} = \Delta D_F^2 / D_F^2 .$$

Now, we find the relation between the variation of distortion and the variation of rate. Again, the theorem of equal merits provides

$$\forall \text{coded } n \quad M_\bullet \Delta R_{n,\bullet} = \Delta D_{n,\bullet}^2$$

and we find

$$\Delta D_F^2 = \sum_\bullet \rho_\bullet \Delta D_{p,\bullet}^2 = \sum_\bullet \rho_\bullet M_\bullet \Delta R_\bullet = \sum_\bullet \rho_\bullet M_\bullet \Delta R_{S,\bullet} / N_\bullet .$$

Then, the variation PSNR can be restated as

$$D_F^2 \, \Delta\mathrm{PSNR} = \Delta D_F^2 = \sum_\bullet \rho_\bullet M_\bullet \Delta R_{S,\bullet} / N_\bullet = M_F \sum_\bullet \rho_\bullet \Delta R_{S,\bullet} = M_F \Delta R_F$$

and we define **the video merit** as the PSNR merit as follows.

$$M_{Video} := \frac{\Delta\mathrm{PSNR}}{\Delta R_F} = M_F / D_F^2$$

The video merit is a constant provided to the encoder. It replaces the parameter QP used in standard codecs. We just have proven the implicit relation between video merit and frame merit.

$$\boxed{M_{Video} D_F^2 = M_F}$$

2.3.7.2 Case of a video with the three components YUV

In case of colored video with three components YUV, it is common to judge the compression capability by looking at the total rate, i.e. the rate of Y+U+V, and the separate average PSNR of the three components.

We deduce that the evaluation function of the quality must be additive in each of the three PSNR. Unfortunately, there is still no clear theory to find out a "good" and systematic balance between the qualities of the three components. Usually the balance is controlled via a single QP, applied to the luma component, and the QP of the chroma is then derived using a non-adaptive relation. In order to better control this balance, two new parameters θ_U and θ_V are introduced to adjust the coding of the chrominance in order to obtain balances U/Y and V/Y close to what is found by a HEVC-based encoder. This will allow fair comparisons with other HEVC-based proposals.

The video quality function is stated as

$$Q(R_{F,Y}, R_{F,U}, R_{F,V}) := \mathrm{PSNR}_{\mathrm{Y}} + \theta_U \mathrm{PSNR}_{\mathrm{U}} + \theta_V \mathrm{PSNR}_{\mathrm{V}}$$

with θ_U and θ_V fixed for the whole encoding of a video sequence. It is a function of the three rates of the components. From the calculations of the preceding section, for the case with one component only, we deduce the partial derivatives of the quality function.

$$\frac{\partial Q}{\partial R_{F,Y}} = \frac{\partial Q}{\partial R_{F,Y}} = M_{F,Y} / D_{F,Y}^2 \qquad \frac{\partial Q}{\partial R_{F,U}} = \theta_U M_{F,U} / D_{F,U}^2 \qquad \frac{\partial Q}{\partial R_{F,V}} = \theta_V M_{F,V} / D_{F,V}^2$$

Optimal coding means that no component is favoured compared to another one; practically all derivatives must be equal. This common value is the video merit.

$$\boxed{M_{Video} := M_{F,Y} / D_{F,Y}^2 = \theta_U M_{F,U} / D_{F,V}^2 = \theta_V M_{F,V} / D_{F,V}^2}$$

Thus, the codec has for entry three parameters:

- **the video merit** M_{Video}**,**
- **the balancing θ_U of chrominance U,**
- **the balancing θ_V of chrominance V.**

2.3.7.3 Finding the frame merit from the video merit and the θ parameters

To ensure the homogeneity in notations, we set the balancing parameter for the component Y,

$$\theta_Y := 1$$

such that one gets

$$M_{Video} = \theta_* M_{F,*} / D_{F,*}^2$$

for all three components $* \in \{Y, U, V\}$. Finding a component merit is equivalent to finding the rightmost zero of the function

$$e(M_*) := M_{Video} D_{F,*}^2 (M_*) - \theta_* M_* .$$

Actually, the merit zero $M_* = 0$ is a solution and corresponds to a raw video coding. We are not interested in this solution; that is why we talk of the "rightmost" zero.

$$e(M_{F,*}) = 0 \qquad M_{F,*} > 0$$

The function $e(M)$ is continuous. One can see the frame merit $M_{F,*}$ as the intersection of the line

$$M_* \mapsto \theta_* M_* / M_{Video}$$

and the function of the distortion

$$M_* \mapsto D_{F,*}^2 (M_*)$$

which is concave. This ensures that there is at most two solutions (which means the intersection set of a line and a concave function), but we know that the origin is solution, so there is only one strictly positive solution; this solution is the frame merit that we look for.

We use a simple dichotomy scheme to find the rightmost zero of the function $e(M_*)$. This automatically excludes the raw video coding solution by taking a strictly positive lower bound for M_* at the starting point of the algorithm. Basically, the algorithm is as below

1. set initial M_{low} and M_{high}
2. take $M_{middle} = (M_{low}+M_{high})/2$
3. depending on the sign of $e(M_{middle})$, update bounds M_{low} and Mhigh
4. loop to 2. until convergence

2.3.7.4 Quality and rate control

In the previous section, it has been shown how to encode a frame for a given video merit. By changing the function e to

$$e(M_*) := D^2_{F,*}(M_*) - D^2_t$$

where D^2_t is a targeted distortion, one ensures the encoding of all frames at a common mean distortion D^2_t.

Similarly, one can control the rate of a frame by using the function

$$e(M_*) := R_{F,*}(M_*) - R_t$$

where R_t is the targeted rate.

2.3.8 Quantization and entropy coding

2.3.8.1 Choice of optimal quantizers

Once the segmentation into block types is obtained, DCT transforms of each block, adapted to the underlying size of the block type, is computed. The statistics of each DCT channel are computed and modeled by GGD's.

Video merit M_{Video} and chrominance balancing parameters θ_U, θ_V are provided as input to the codec and the frame merits M_Y, M_U and M_V are determined. Then, for each component, for each block type and for each DCT channel, the target distortion $D^2_{n,\bullet,*}$ is determined thanks to the theorem of encoding with equal merits and the formula

$$M_{\bullet,*} = \frac{2D^2_{n,\bullet,*}}{f'(-\ln(D_{n,\bullet,*}/\sigma_{n,\bullet,*}))}$$

The functions f', which depend on the value of the modeled parameter $\beta_{n,\bullet,*}$, are computed and tabulated off-line. This implicit equation in $D^2_{n,\bullet,*}$ is solved by using a dichotomy scheme. A fixed-point scheme could also be used.

Then, one chooses the best optimal quantizer $Q_{n,\bullet,*}$ associated to the distortion $D^2_{n,\bullet,*}$. For instance one may select, from a list of optimal quantizers corresponding to the associated parameter $\beta_{n,\bullet,*}$, the quantizer with least rate among quantizers having a distortion less than the targeted distortion $D^2_{n,\bullet,*}$. Alternatively, an interpolation between quantizers of the list may be performed to increase the precision of the selected quantizer.

The quantization is performed by the chosen quantizers to obtain the quantized data of transformed coefficients. Practically, these data are just symbols corresponding to the index of the quantum (or interval or Voronoi cell in 1D) in which the value of the coefficient falls in.

2.3.8.2 Entropy coding

Finally, the quantized data are coded by a **context-free arithmetic coding** following the statistical distribution of the DCT channels. No context is needed because the probability of occurrence of each quantum is known a priori thanks to the knowledge of the GGD. These probabilities are computed off-line and stored within the data associated to each quantizer.
Context free coding also allows a straightforward design of the codec with the so-called "random spatial access" feature.

2.3.9 Block type competition for segmentation optimization

As already pointed out, the efficiency of the method described in previous sections, is sensitive to the initial block type segmentation. The initial segmentation method described previously is not optimal and is improved by using a competition process between block types.

2.3.9.1 Iterative improvement of the segmentation

The principle of the competition is simple: one starts from an initial segmentation (found from the morphology of the residual, as already described) into block types and tries to improve it iteratively by evaluating the cost of a change of block type for each block. Choosing the best block type from the cost evaluations will be called "block type competition" hereafter.
In order to decide which block type is the best for the compression of the data, an objective criterion of evaluation is needed; we will use a variation of the Lagrangian cost $D^2+\lambda R$ used in many codecs.
The competition process may be summarized as follows:

1. start from an initial segmentation into block types
2. compute GGD statistics for each DCT coefficient of each block type
3. compute the GGD models for each DCT coefficient of each block type
4. for each component YUV, determine a temporary merit of the frame, temporary merits of block types, temporary merits of encoding of DCT channels and temporary quantizers
5. for each block of the frame, perform a Lagrangian cost competition between block types for this block; the cost is computed from
    a. the bit-rate needed for encoding Y,U,V by using the temporary quantizers of the competing block type
    b. and the total YUV distortion after quantization and dequantization by using the temporary quantizers of the competing block type;

    and change the type of the block into the type with the best (=smallest) cost
6. loop to 2. until a fixed number of iteration is reached or until the segmentation does not evolve anymore
7. by using the last segmentation computed by the iterative process, compute GGD statistics and models for each DCT coefficient of each block type
8. for each component YUV, determine the actual coding merit of the frame, merits of block types, merits of encoding of DCT channels and quantizers used for encoding the frame
9. quantize and encode (by using a context free arithmetic entropic encoder) the coefficients of the DCT channels of all blocks

The computation of GGD statistics needs to be located inside the loop because, after the first iterations, the statistics are not consistent anymore after having performed block type competition. However, after a small number of iterations (typically from 5 to 20), one observes a convergence of the iterative process to a local optimum for the segmentation.

2.3.9.2 General formulation of the Lagrangian cost

It turns out that, instead of using the formulation $D^2+\lambda R$, it is simpler to use the formulation $D^2/\lambda+R$ when dealing with several color components. Again, it is supposed that one encodes a frame made of three YUV components and, as described above, the components are encoded separately with their own merits and quantizers. However, there is a coupling between components because there is **only one quad-tree for the three components**. Of course, the competition described below is not limited to only one quad-tree and may be adapted in a case of several quad-trees dedicated to one component each.

One can put the cost under the very general form

$$C_\bullet = \frac{D_{\bullet,Y}^2}{\lambda_{\bullet,Y}} + \frac{D_{\bullet,U}^2}{\lambda_{\bullet,U}} + \frac{D_{\bullet,V}^2}{\lambda_{\bullet,V}} + R_{\bullet,Y} + R_{\bullet,U} + R_{\bullet,V} + R_{\bullet,QT}$$

where

- $C_\bullet$ is the cost of the block type for the selected block
- $D_{\bullet,Y}^2$ is the luminance distortion of the selected block after quantization and dequantization
- $D_{\bullet,U}^2$ is the chrominance distortion U of the selected block after quantization and dequantization
- $D_{\bullet,V}^2$ is the chrominance distortion V of the selected block after quantization and dequantization
- $R_{\bullet,Y}$ is the bit-rate generated by the encoding of the luminance of the selected block
- $R_{\bullet,U}$ is the bit-rate generated by the encoding of the chrominance U of the selected block
- $R_{\bullet,V}$ is the bit-rate generated by the encoding of the chrominance V of the selected block
- $R_{\bullet,QT}$ is the bit-rate associated to the parsing of the generalized quad-tree to mark the type of the selected macroblock.

The three Lagrange's parameters $\lambda_{\bullet,Y}$, $\lambda_{\bullet,U}$, $\lambda_{\bullet,V}$ are computed from the geometrical characteristics of the block types and the merits of encoding M_Y, M_U and M_V.

2.3.9.3 Lagrange's parameter for the luminance

The cost of encoding for the luminance can be written

$$C_{\bullet,Y} = \frac{D_{p,\bullet,Y}^2}{\lambda} + R_{\bullet,Y}$$

where the Lagrange's parameter is given by

$$\lambda = -\frac{\partial D_{p,\bullet,Y}^2}{\partial R_{\bullet,Y}}.$$

From the definition of the block merit and the size of the block, we get $\Delta D_{p,\bullet,Y}^2 = M_{\bullet,Y}\Delta R_{\bullet,Y}$ and $M_{\bullet,Y} = N_{\bullet,Y}M_{F,Y}$ such that we deduce

$$\lambda = -\frac{\partial D_{p,\bullet,Y}^2}{\partial R_{\bullet,Y}} \approx \frac{\Delta D_{p,\bullet,Y}^2}{\Delta R_{\bullet,Y}} = M_{\bullet,Y} = N_{\bullet,Y}M_{F,Y}$$

and the luminance cost becomes

$$C_{\bullet,Y} = \frac{D_{p,\bullet,Y}^2}{N_{\bullet,Y}M_{F,Y}} + R_{\bullet,Y} + R_{\bullet,QT}.$$

2.3.9.4 Lagrange's parameter for the chrominance

The cost of encoding for the chrominance U can be written

$$C_{\bullet,U} = \frac{D_{p,\bullet,U}^2}{\lambda} + R_{\bullet,U}$$

where the Lagrange's parameter is given by

$$\lambda = -\frac{\partial D_{p,\bullet,U}^2}{\partial R_{\bullet,U}}.$$

Again, we finally get a closed form for the Lagrange's parameter

$$\lambda = -\frac{\partial D_{p,\bullet,U}^2}{\partial R_{\bullet,U}} \approx \frac{\Delta D_{p,\bullet,U}^2}{\Delta R_{\bullet,U}} = M_{\bullet,U} = N_{\bullet,U}M_{F,U}.$$

The chrominance cost for U can be rewritten

$$C_{\bullet,U} = \frac{D^2_{p,\bullet,U}}{N_{\bullet,U} M_{F,U}} + R_{\bullet,U}$$

under the assumption that there is no quad-tree cost because the chrominance segmentation is inferred by the luminance segmentation. One gets a similar cost for the component V.

2.3.9.5 Combined luminance and chrominance cost

The combined cost, taking into account both luminance and chrominance, is simply the sum of the three underlying Y,U,V costs

$$C_{\bullet,YUV} = \frac{D^2_{p,\bullet,Y}}{N_{\bullet,Y} M_{F,Y}} + \frac{D^2_{p,\bullet,U}}{N_{\bullet,U} M_{F,U}} + \frac{D^2_{p,\bullet,V}}{N_{\bullet,V} M_{F,V}} + R_{\bullet,Y} + R_{\bullet,U} + R_{\bullet,v} + R_{\bullet,QT}\,.$$

The cost of the quad-tree syntax is estimated directly from the entropy associated to the probabilities $P(L|s_B)$ of occurrence of block types.

2.3.9.6 Practical computation of the cost

The distortions $D^2_{p,\bullet,*}$ are computed by using the quantizers associated to the block type, then by applying the associated dequantization and finally by comparing the result with the original residual. This last step can be done in the DCT transform domain because the IDCT is a L_2 isometry: total distortion in the DCT domain is the same as the total pixel distortion (practically up to a scaling factor).

Bit-rates $R_{\bullet,*}$ are estimated without performing the entropy encoding of the quantized coefficients. Actually, one knows the mean rate of each quantum of the quantizers; this rate is simply computed from the probability of a coefficient to fall into this quantum; this probability is provided directly by the integral of the GGD channel model on the quantum.

2.3.9.7 Effect of the block size on the cost

The size (more precisely the surface) of a block impacts the cost formula through the geometrical parameters $N_{\bullet,*}$.

For instance, let us consider the case of a 32x32-pixel unit area and a 4:2:0 YUV color format.

The cost of a 32x32 block becomes

$$C_{32,YUV} = \frac{D^2_{p,32,Y}}{M_{F,Y}} + \frac{D^2_{p,32,U}}{4M_{F,U}} + \frac{D^2_{p,32,V}}{4M_{F,V}} + R_{32,Y} + R_{32,U} + R_{32,V} + R_{32,QT}$$

because $N_{32,Y} = 1$ and $N_{32,U} = N_{32,V} = 4$. This last value comes from the fact there are four times less pixels in U and V compared to Y when dealing with 4:2:0 videos.

Similarly, one finds the cost of a 16x16 block

$$C_{16,YUV} = \frac{D^2_{p,16,Y}}{4M_{F,Y}} + \frac{D^2_{p,16,U}}{16M_{F,U}} + \frac{D^2_{p,16,V}}{16M_{F,V}} + R_{16,Y} + R_{16,U} + R_{16,V} + R_{16,QT}$$

and the cost of a 8x8 block

$$C_{8,YUV} = \frac{D^2_{p,8,Y}}{16M_{F,Y}} + \frac{D^2_{p,8,U}}{64M_{F,U}} + \frac{D^2_{p,8,V}}{64M_{F,V}} + R_{8,Y} + R_{8,U} + R_{8,V} + R_{8,QT}$$

2.3.9.8 Bottom-to-top competition

For a 32x32 area to be coded, one has to decide both

- the segmentation of this area into 32x32, 16x16 and 8x8 blocks,
- the choice of the label for each block,

such that the cost is minimized. This may lead to a very big number of possible configurations to evaluate. Fortunately, by using the classical so-called bottom-to-top competition technique, one dramatically decreases the number of configurations to evaluate.

The fundamental tool is the additivity of costs. For instance, a 16x16 block may be segmented into four 8x8 blocks. By using 8x8 cost competition, we determine the best type (e.g. the most competitive type which gives the type with the smallest cost) for each of the four 8x8 blocks. Then, the cost associated with the 8x8 segmentation is just the addition of the four underlying best 8x8 costs.

Of course, the additivity of the costs does not apply to the cost of the quad-tree parsing but only to the D^2/λ and R parts of the cost; the quad-tree cost is calculated from the segmentation.
From there, one starts the bottom-to-top process by comparing this best 8x8 cost for the 16x16 block to the costs of 16x16 block types. Now, if we supposed that there are two 16x16 block types, one has to compare three costs:

- the best 8x8 cost deduced from cost additivity,
- the 16x16 cost from 16x16 label 1,
- the 16x16 cost from 16x16 label 2.

The smallest cost decides the segmentation and the types of the 16x16 block. Then, one pushes the bottom-to-top process to a bigger scale: once the segmentation and types of 16x16 block has been decided, one can use competition to decide the segmentation of 32x32 blocks by performing a competition between the best 16x16 costs (again by using additivity) and 32x32 costs of 32x32 block types.
One understands easily that the bottom-to-top competition is not limited to three different sizes, not even to square blocks.

2.3.10 Reduction of the cost of statistics

2.3.10.1 Encoding of statistics

By using the theorem of equal merits of encoding, the encoder determines which DCT channels must be coded or not. Let us define the characteristic function of encoding

$$\chi_{n,\bullet,*} \in \{0,1\}$$

Its value is 1 if the associated DCT channel is encoded, i.e. $D_{n,\bullet,*} < \sigma_{n,\bullet,*}$ is found by the encoder, and 0 otherwise. Of course, if a channel is not encoded, there is no need (it would be a waste of bit-rate) to send the associated statistics.

- if $\chi_{n,\bullet,*} = 1$, associated statistics $\sigma_{n,\bullet,*}$ and $\beta_{n,\bullet,*}$ are encoded in the stream
- if $\chi_{n,\bullet,*} = 0$, associated statistics are not encoded in the stream

The parameter $\beta_{n,\bullet,*}$ is tabulated on 8 values as follows

$$\beta_{n,\bullet,*} \in \{0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$$

and is encoded on 3 bits through a look-up table. The parameter $\sigma_{n,\bullet,*} > 0$ is quantized up to a fixed precision $\sigma_{0,*}$ such that

$$\sigma_{n,\bullet,*} = \hat{\sigma}_{n,\bullet,*}\sigma_{0,*} \quad \text{with } \hat{\sigma}_{n,\bullet,*} \in \mathrm{N}^{\times},$$

and the integer $\hat{\sigma}_{n,\bullet,*}$ is encoded in the bit-stream. The number of bits $N^{\sigma}_{\bullet,*}$ needed to encode the $\hat{\sigma}_{n,\bullet,*}$'s depends on the block type and the component. It is set to be enough to encode the maximum value taken on all the $\hat{\sigma}_{n,\bullet,*}$'s.

$$N^{\sigma}_{\bullet,*} = \left\lceil \log_2\left(\max_{n;\chi_{n,\bullet,*}=1} \hat{\sigma}_{n,\bullet,*} \right) \right\rceil + 1$$

The number of encoded channels for a component of a block type is

$$N^{\chi}_{\bullet,*} = \sum_n \chi_{n,\bullet,*} .$$

Now, we are ready to describe the practical encoding of the statistics of the component $*$ of the block type $\bullet$.

1. encode $N^{\chi}_{\bullet,*}$ (on 10 bits for 32x32 blocks with at most 1024 channels; 8 bits for 16x16 blocks, and so on)
2. if $N^{\chi}_{\bullet,*} = 0$, encoding is finished
3. encode $N^{\sigma}_{\bullet,*}$ on 4 bits (this number of bits depends on the scaling of the DCT and the precision $\sigma_{0,*}$)
4. loop on DCT channels, i.e. loop on n

a. encode $\chi_{n,\bullet,*}$ on 1 bit

b. if $\chi_{n,\bullet,*}=1$, encode $\hat{\sigma}_{n,\bullet,*}$ on $N^{\sigma}_{\bullet,*}$ bits and $\beta_{n,\bullet,*}$ on 3 bits

c. until the number of encoded channels reaches $N^{\chi}_{\bullet,*}$

The decoding process is straightforward

1. decode $N^{\chi}_{\bullet,*}$
2. if $N^{\chi}_{\bullet,*}=0$, decoding is finished
3. decode $N^{\sigma}_{\bullet,*}$
4. loop on DCT channels, i.e. loop on n
   a. decode $\chi_{n,\bullet,*}$
   b. if $\chi_{n,\bullet,*}=1$, decode $\hat{\sigma}_{n,\bullet,*}$ and $\beta_{n,\bullet,*}$
   c. until the number of decoded channels reaches $N^{\chi}_{\bullet,*}$

The loop on n may follow a so-called zigzag scan of the DCT coefficients to allow a faster reaching of the bound $N^{\chi}_{\bullet,*}$ and saving the rate of many potentially useless flags $\chi_{n,\bullet,*}$.

2.3.10.2 Keeping statistics for several frames: principles

Encoding the statistics every frame may not be optimal because it often happens that frames f and f+1 have contents similar enough such that statistics of the frame f may be used to encode the frame f+1. To do so, one needs to

- find a process to keep useful statistics from a frame f, f-1, f-2,.. to be used in frame f+1
- find a criterion which decides when it is time to trough away statistics and compute new ones for the current frame.

In the present version of the codec, there are two types of frames

- "re-stat" frames for which all statistics concerning that frame only are encoded in the stream with the method described in the previous section
- "non re-stat" frames for which all available statistics coming from the preceding frames are used and some additional statistics may be encoded.

In non re-stat frames the available statistics come from

- the latest re-stat frame
- the union of additional statistics provided by the frames between the latest re-stat frame and the current frame (included)

The additional statistics are not computed on the current non re-stat frame. Actually, the statistics are computed only on re-stat frames. The additional statistics are just the addition of channels which were not used in the preceding frames but are now useful in the current non re-stat frame; the statistics of these added channels are those computed on the latest re-stat frame.
The use of **non re-stat frames** allows not only the **saving of bit-rate** thanks to a smaller rate for statistics in the bit-stream, but also **accelerates the convergence of the competition process**. Actually, when statistics are kept constant in the iterative competition process of the encoder, this iterative process converges much faster; typically in 2-3 iterations compared to 5-20 iterations for re-stat frames.

The criterion to decide to re-stat a frame I is based on the variation of the frame distortion $D^2_{F,Y}$. If the relative variation of the distortion of the current frame compared to the distortion of the last re-stat frame is above a given threshold (say 3%), then it is decided to re-stat next frame.

2.3.10.3 Keeping statistics for several frames in practice

A flag is encoded in the header of each frame to precise if the frame is "re-stat" or "non re-stat".
For re-stat frame, the encoding of statistics is performed as described above. Statistics of all channels (even channels not used for re-stat frame) are kept for possible future use in non re-stat frames, until a new frame is re-stated.
We introduce a new characteristic function

$$\mathrm{X}_{n,\bullet,*} \in \{0,1\}$$

whose value is 1 if the associated statistics $\sigma_{n,\bullet,*}$ and $\beta_{n,\bullet,*}$ have already been encoded in the bit-stream since a frame has been re-stated. By convention, we set

$$\mathrm{X}_{n,\bullet,*} = 0$$

for re-stated frames. For a non re-stat frames f, one writes

$$\mathrm{X}^{f}_{n,\bullet,*} = \max_{f_r \le f' < f} \chi^{f'}_{n,\bullet,*}$$

where f_r is the index of the last re-stat frame before frame f. We added the obvious superscript f' to $\chi_{n,\bullet,*}$ to signify that this characteristic function applies to the frame f'. Obviously, one can rewrite it under an inducted form.

$$\mathrm{X}^{f}_{n,\bullet,*} = \max\left\{\mathrm{X}^{f-1}_{n,\bullet,*}, \chi^{f-1}_{n,\bullet,*}\right\} \qquad (f > f_r)$$

This is important because it will allow to determine $\mathrm{X}^{f}_{n,\bullet,*}$ just by the decoding of the statistics and without using the effective computation of the $\chi_{n,\bullet,*}$ on the decoder side.

The practical encoding of statistics for a non re-stat frame f is performed as follows

1. encode $N^{\chi}_{\bullet,*}$ (the number of additional needed statistics)
2. if $N^{\chi}_{\bullet,*} = 0$, encoding is finished
3. encode $N^{\sigma}_{\bullet,*}$
4. loop on DCT channels, i.e. loop on n
   a. if $X^{f}_{n,\bullet,*} = 1$, set $X^{f+1}_{n,\bullet,*} = 1$ and do nothing (statistic already available)
   b. if $X^{f}_{n,\bullet,*} = 0$,
      i. encode $\chi_{n,\bullet,*}$ on 1 bit
      ii. if $\chi_{n,\bullet,*} = 1$,
         - encode $\hat{\sigma}_{n,\bullet,*}$ on $N^{\sigma}_{\bullet,*}$ bits and $\beta_{n,\bullet,*}$ on 3 bits (additional encoded channel)
         - set $X^{f+1}_{n,\bullet,*} = 1$
      iii. if $\chi_{n,\bullet,*} = 0$,
         - set $X^{f+1}_{n,\bullet,*} = 0$ (channel still not available)
   c. until the number of additional encoded channels reaches $N^{\chi}_{\bullet,*}$

The decoding process is straightforward. Note that the decoding of flags $\chi_{n,\bullet,*}$ allows updating $X^{f+1}_{n,\bullet,*}$ for the following frame without starting the decoding of the frame. This means that there is no dependence between frames and the method using no re-stat frames can be used between Intra frames. In the following section, we show how to use carriers of statistics to be able to decode a non re-stat Intra frame without decoding the preceding frames.

2.3.10.4 Carriers for statistics

Non re-stat frames refine the set of statistics parameters of the previous re-stat frame by defining new statistics parameters that have not been encoded for the re-stat frame. When performing random access, statistics parameters of the immediate previous re-stat frame must be decoded prior to the accessed frame.
To avoid introducing dependency between VCL NAL units, statistics are encoded in a non VCL NAL unit that is transmitted in addition to the VCL NAL units. Consequently, only non VCL NAL units that correspond to the re-stat frame's statistics are required before being able to decode the random access point.
Statistics are defined in the Adaptation Parameter Sets NAL unit: each slice NAL unit refers to one APS NAL unit with an identifier stored in the slice header. APS includes a flag that states if the frame is a re-stat or a non re-stat frame. If the flag is set to 1, the APS NAL unit includes all statistics parameters required for decoding slice NAL units which refer to this APS. On the other hand if the flag is equal to 0, the APS NAL unit includes refinement of the statistics in previous APS parameter.

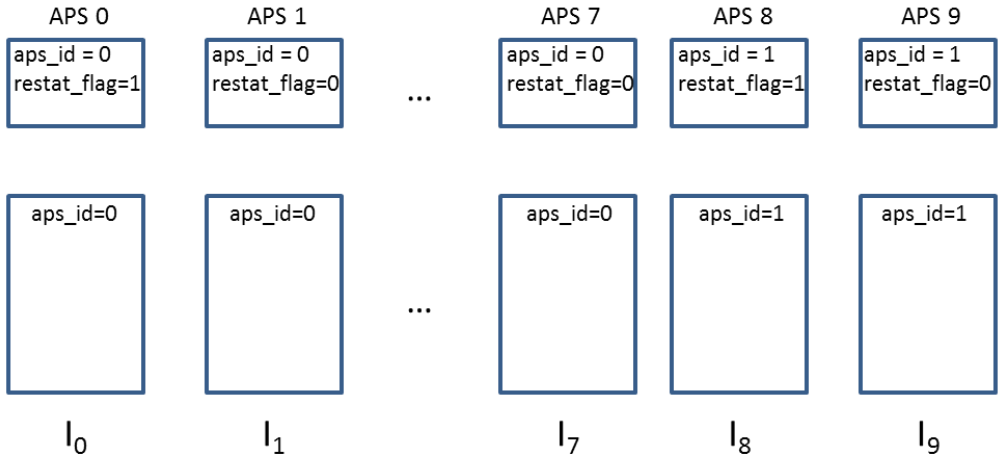

Figure 16: Set of frames with associated APS NAL units

For example, the video frames of the Figure 16 are associated with an APS NAL unit. The frame I_0 and I_8 are re-stat frames since associated to APS NAL units which have a restat_flag set to 1. On the contrary the other frames reference APS NAL units with restat_flag set to 0 and are thus non re-stat frames.
When decoding one APS NAL unit which has a restat_flag equal to 1 all statistics parameters are reset and replaced by the values defined in the APS NAL unit's data. Following APS NAL units which have the same aps_id refines the statistics of this APS NAL unit. It is thus recommended to increase the aps_id when a new set of statistics are used i.e. the APS NAL unit has a restat_flag equal to 1. This recommendation eases the identification of APS NAL units which define one set of statistics.
When randomly accessing to one frame, all APS NAL units which have an aps_id equal to the aps_id of the first accessed slice NAL unit, must be decoded prior to slice data.

2.3.11 Lambda parameters, deduced from the merits, for post-filtering

The texture coding process provides three merits $M_{F,Y}$, $M_{F,U}$ and $M_{F,V}$. They are also available on the decoder side because they are encoded in the bit-stream.
It is also common to post-process a decoded frame by applying one or several filters like a de-blocking filter (DBF), a Sample Adaptive Offset filter (SAO) or an Adaptive Loop Filter (ALF). The DBF must know the QP parameter to be applied on both encoder and decoder sides. The SAO and the ALF must be provided a rate-distortion slope (a lambda parameter) to determine their encoding; the slope is not needed on the decoder side.
The lambda parameters are deduced straightforwardly from the merits for each color component.

$$\lambda_* \propto M_{F,*}$$

The constant of proportionality depends on the area unit chosen for calculating the pixel merits. In the present implementation, area unit is a 16x16 block of pixels for the luminance component (8x8 for the two chrominance components) and this leads to the equality

$$\lambda_* = M_{F,*}$$

with the current definition (compliant with the version HM6.1 of HEVC) of the lambda parameters in the SAO and the ALF.
In case only one common slope, for instance in the ALF, is used for both color components U and V, a mean lambda for the chrominance is determined by

$$\lambda_{UV} = 2/(1/M_{F,U} + 1/M_{F,V}).$$

The QP parameter used by the DBF filter is also deduced from the luminance merit by using a high rate asymptotic approximation on uniform quantizers.

$$\mathrm{QP}_Y = 3\log_2(M_{F,Y}) + 9$$

and the same QP is used for the DBF of chrominance components.

$$\mathrm{QP}_U = \mathrm{QP}_V = \mathrm{QP}_Y.$$

2.4 Inter enhancement pictures coding/decoding process

This section describes the compression system for Inter pictures (namely P or B pictures) in the proposed scalable HEVC extension. As introduced in sections 2.1.3 and 2.1.4, the coding/decoding of Inter pictures involves the HEVC Inter coding modes, together with additional so-called Inter-layer prediction modes. Inter-layer prediction tools are described in section 2.4.1. Note that the difference intra coding mode is taken from [4] as is, hence is not detailed.

The encoder control methods include a rate distortion optimized coding mode selection similar to that of the HM6.1, and a picture QP setting that is fixed as a function of the picture index and temporal level. These strategies are described in section

2.4.1 Inter-layer prediction modes

The Intra and temporal prediction system specified by the HEVC specifications of HM6.1 has been extended to support Inter-layer prediction tool. The goal of Inter-layer prediction is to exploit the redundancy that exists between the coded base layer and the enhancement pictures to code, in order to obtain as much compression efficiency as possible in the enhancement layer. The following of this section describes all Inter-layer prediction tools included in the CSC proposal. Each of this Inter-layer prediction mechanism is performed the same way at the encoder and the decoder sides.

2.4.1.1 "IntraBL" prediction mode

IntraBL Inter-layer prediction simply consists in predicting a Coding Unit from its co-located area in the reconstructed, optionally up-sampled, base picture. The Intra BL mode is allowed whatever the way the co-located base CU(s) of a given enhancement was (were) coded, thanks to the multiple loop decoding approach chosen.

The Intra BL coding mode is signaled at the PU level as a particular Inter-layer prediction mode.

2.4.1.2 "BaseMode" prediction mode

The "BaseMode" prediction tool consists in predicting an entire Coding Unit from its co-located spatial area in the so-called Base Mode prediction picture. The Base Mode prediction picture is constructed in an identical way on the encoder and decoder sides, with the help of some prediction information derived from the base layer.

In the case of SNR scalability, this inferred prediction information corresponds to the Coding Unit structure of the base picture, taken as is, before the motion information compression step performed in the base layer. In the case of spatial scalability, the base layer prediction information first undergoes a so-called prediction information up-sampling process, which is described in section 2.4.1.2.1.

Once the derived prediction information is obtained, the Base Mode prediction picture is computed, through temporal prediction of derived Inter CUs and Intra BL prediction of derived Intra CUs (see section 2.4.1.2.3).

Prediction information inheritance in case of SNR scalability

In case of SNR scalability, the Inter-layer derivation of prediction information is trivial. The derived prediction information corresponds to the prediction information of the coded base picture.

2.4.1.2.1 Inter-layer prediction of CU/TU/PU partitions for the dyadic case

Figure 17 depicts the prediction information up-sampling process, executed both by the encoder and the decoder in order to construct the "Base Mode" prediction picture for the dyadic case (spatial ratio is two).

The left side of Figure 17 illustrates a part of the base layer picture. In particular, the Coding Unit representation that has been used to encode the base picture is illustrated, for the two first LCU (Largest Coding Unit) of the picture. The Coding Unit quad-tree representation of the second LCU is represented, as well as their PU partitions. Moreover, the motion vector associated to each PU is showed.

On the right side of Figure 17, the result of the prediction up-sampling process can be seen, in case of dyadic spatial scalability. On this figure, the LCU size is the same in the enhancement picture and in the base picture. As can be seen, the up-sampled version of base LCU 1 results in the enhancement LCUs 2, 3, 6 and 7. Moreover, the Coding Unit quad-tree has been re-sampled as a function of the scaling ratio that exists between the enhancement picture and the base picture. The Prediction Unit partitioning is of the same type (i.e. PU's have the same shape) in the enhancement layer and in the base layer. Finally, motion vector coordinates have been re-scaled as a function of the spatial ratio between the two layers.

In other words, three main steps are involved in the prediction information up-sampling process.

- The Coding Unit quad-tree representation is first up-sampled. To do so, the depth parameter of the base Coding Unit is decreased by 1 in the enhancement layer.

- The Coding Unit partitioning mode is kept the same in the enhancement layer, compared to the base layer. This leads to Prediction Units that have an up-scaled size in the enhancement layer, and have the same shape as their corresponding PU in the base layer.
- The motion vector is re-sampled to the enhancement layer resolution, simply by multiplying their *x* and *y* coordinates by the appropriate scaling ratio.

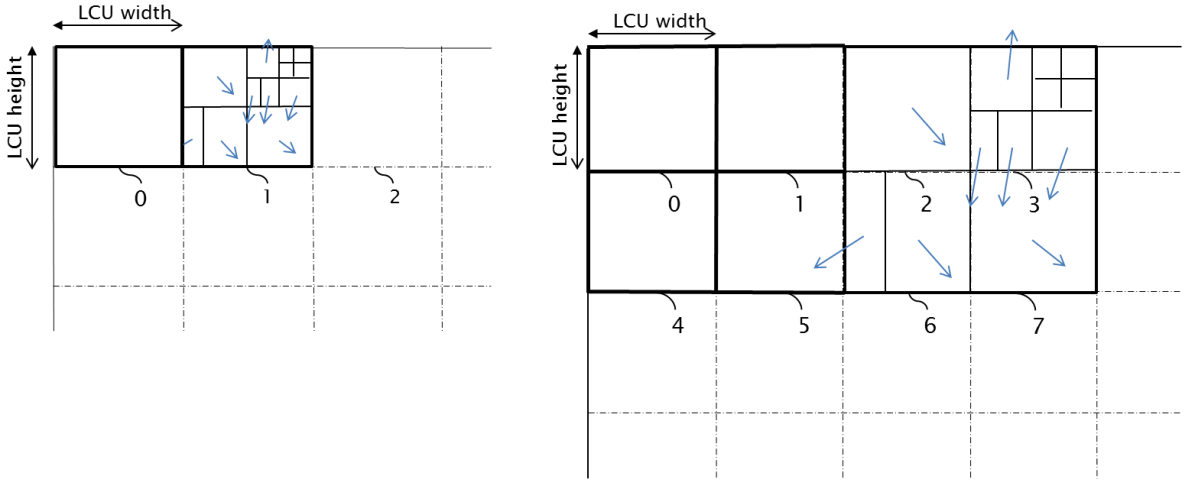

Figure 17: Prediction information up-sampling in dyadic case

As a result of the prediction information up-sampling process, some prediction information is available on the encoder and on the decoder side, and can be used in various Inter-layer prediction mechanisms in the enhancement layer.

In the current scalable encoder and decoder architectures, the up-scaled prediction information is used in two ways.

- It is used in the construction of the "Base Mode" prediction picture of current enhancement picture.
- The up-sampled prediction information is also used for the Inter-layer prediction of motion vectors in the coding of the enhancement picture. Typically, one additional predictor is used compared to HEVC, in the predictive coding of motion vectors. See section 2.4.1.4 for more details.

The overall prediction up-sampling process of Figure 17 consists in up-sampling first the Coding Unit structure, and then in up-sampling the Prediction Unit partitions.

With respect to Coding Unit depth information, Figure 18 illustrates how the HEVC Coding Units are up-sampled from a base layer towards a spatial enhancement layer, in the case of dyadic spatial scalability.

Each Coding Unit has an associated depth level in the quad-tree representation of base LCUs. The up-sampling of Coding Units consists in the following method. For a given Largest Coding Unit (LCU) in the enhancement layer, the Coding Unit that spatially corresponds to that enhancement LCU is searched in the base picture. The enhancement LCU is then given Coding Unit depths values that are equal to the depth values contained in the found base Coding Units, decreased by 1. These decreased depth values then provide a quad-tree that corresponds to the quad-tree in the base picture, up-sampled by 2 in width and height. In case the base CU depth is equal to 0, then the derived CU depth is also equal to zero (same LCU size in base and enhancement layers).

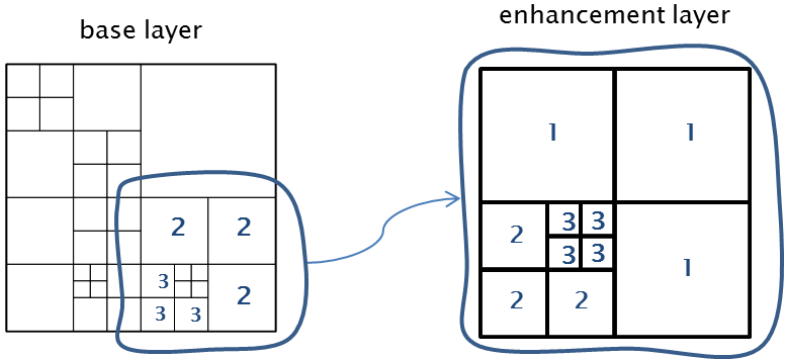

Figure 18: CU depth information in case of dyadic up-sampling

Concerning the up-sampling of PUs partition type in dyadic case, the Inter-layer derivation is straightforward in the general case, and simply consists in providing the enhancement CU with the same PU partition as the co-located base CU.

However, **special care has to be taken in case AMP (Asymmetric Motion Partitioning) is used in the base layer**. AMP is not employed in the CFP simulcast anchors. However, since AMP has been re-introduced into the HEVC main profile, cases where AMP appears in the base layer will have to be properly handled in scalable extension of HEVC.

Figure 19 illustrates the method proposed in order to up-sample asymmetric vertical Prediction Unit partitions for the dyadic case. The similar method for asymmetric horizontal partition is described in Figure 20.The goal of the derivation process applied on PU partition types is to obtain a partitioning into PUs in the enhancement layer that conforms to that of the base layer, in terms of PUs shapes.

The proposed mechanism up-samples base Prediction Units, according to the following policy. For each up-sampled Coding Unit in the enhancement layer, the following applies.

- If the co-located CU in the base layer has a depth strictly higher than 0, then the Prediction Unit type in the enhancement CU is set equal to the Prediction Unit type in the base layer.
- If the base CU has a depth value equal to 0, i.e. the CU has the maximum allowed size, then the following applies.
  - If the co-located base CU has a symmetric Prediction Unit partition, then the enhancement CU is given the 2Nx2N Prediction Unit type. Such case is illustrated on the left of Figure 19. The reason for that is that the base CU, when up-sampled to the enhancement layer resolution, spatially covers four LCU in the enhancement picture.
  - If the co-located base CU has an asymmetric Prediction Unit type, then the enhancement CU is assigned a Prediction Unit type that depends on its spatial position in the enhancement layer, and on the base asymmetric Prediction Unit. This is illustrated on the middle of Figure 19. Indeed, on the example of Figure 19, the base CU has a PU type equal to nLx2N. The spatial up-sampling of the base CU covers 4 LCUs in the enhancement layer. As the goal of the PU up-sampling is to preserve the spatial geometry of the base layer, the enhancement PU assignment is done as follows. Enhancement LCU's that have an even *x* index (left side of the diagram) are given a symmetric PU type equal to Nx2N. Enhancement LCUs that have an odd *x* index (right side of the diagram) are given the 2Nx2N PU type.

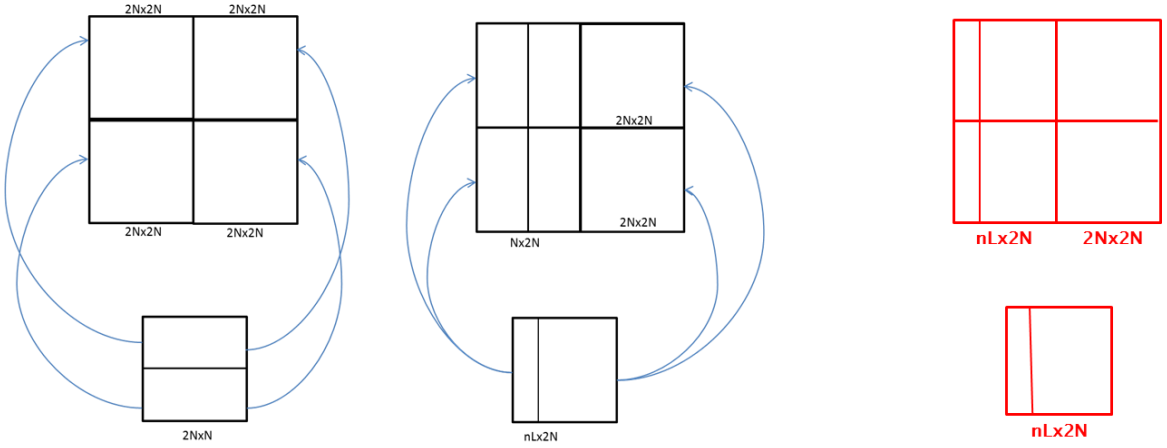

Figure 19: Asymmetric Motion vertical Partition up-sampling for the dyadic case

By comparison the simplest method that could be used is illustrated on the right of Figure 19 and Figure 20. It would consist in systematically providing the enhancement Coding Units with the same Prediction Unit partitions as in the base layer, whatever the enhancement Coding Unit configuration.

The result of our proposed method is that the up-sampled prediction information in our method better represents the motion contained in the considered video sequence.

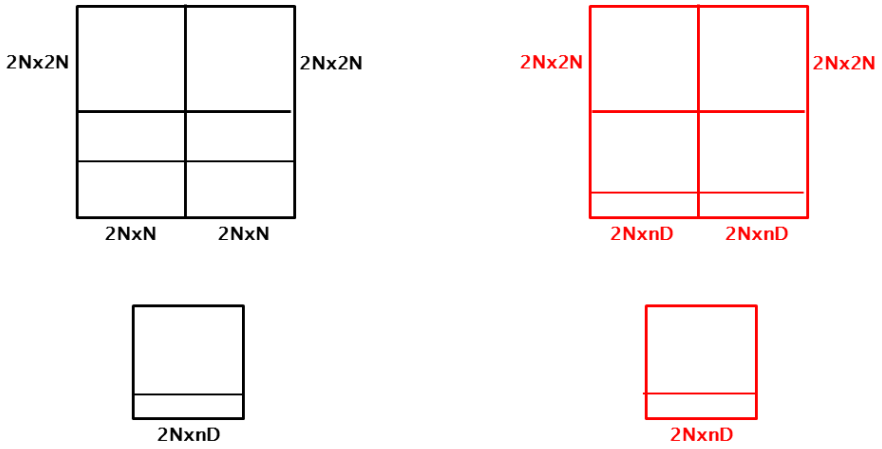

Figure 20: Asymmetric Motion horizontal Partition up-sampling in dyadic case

The policy employed to derive PU type from the base picture to the enhancement picture in the case of dyadic scalability is given by the diagram of Figure 21.

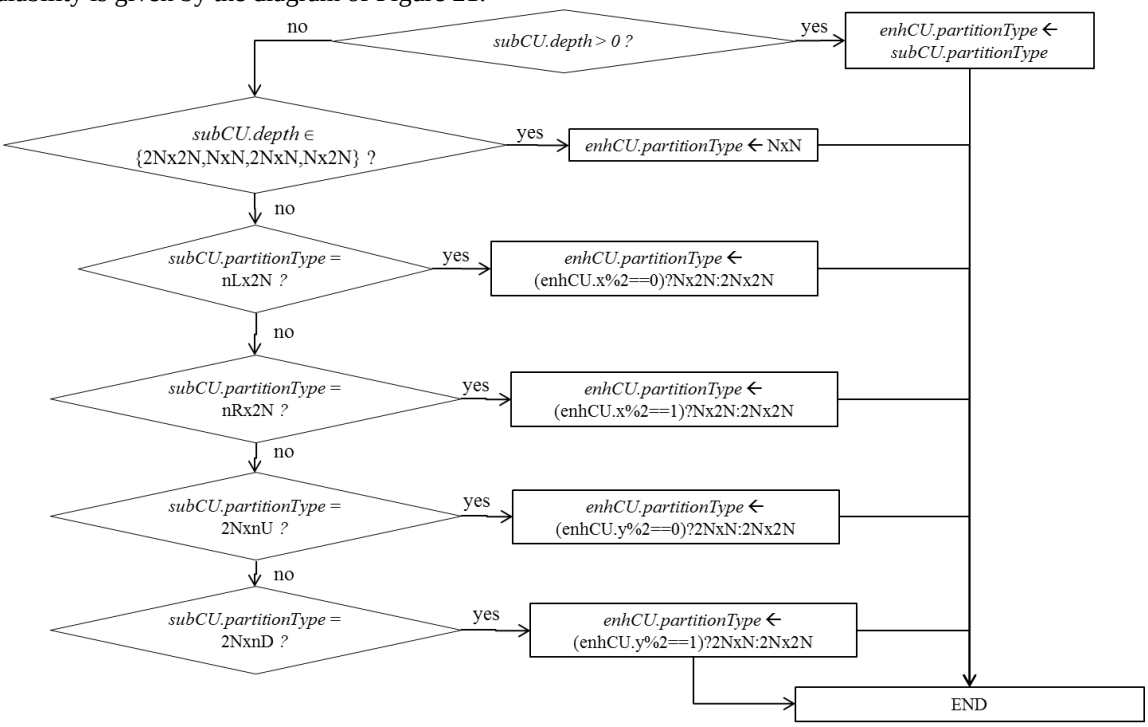

Figure 21: Prediction Unit partitioning derivation process (dyadic case)

2.4.1.2.2 Inter-layer prediction of CU/TU/PU partitions in case of a spatial ratio 1.5

The goal of Inter-layer prediction information derivation is to keep as much accuracy as possible in the up-scaled Prediction Unit and motion information, in order to generate a Base Mode prediction picture which that is as good as possible.

However, in the case of spatial scalability with ratio 1.5, the block-to-block correspondence between the base picture and the enhancement picture differs from the dyadic case. It is illustrated by Figure 23. Therefore, a straight-forward prediction information up-scaling method as that illustrated by Figure 17 does not seem feasible in the case of ratio 1.5, because it would make it very complicated to determine the right CU splitting in the enhancement picture (see Figure 22).

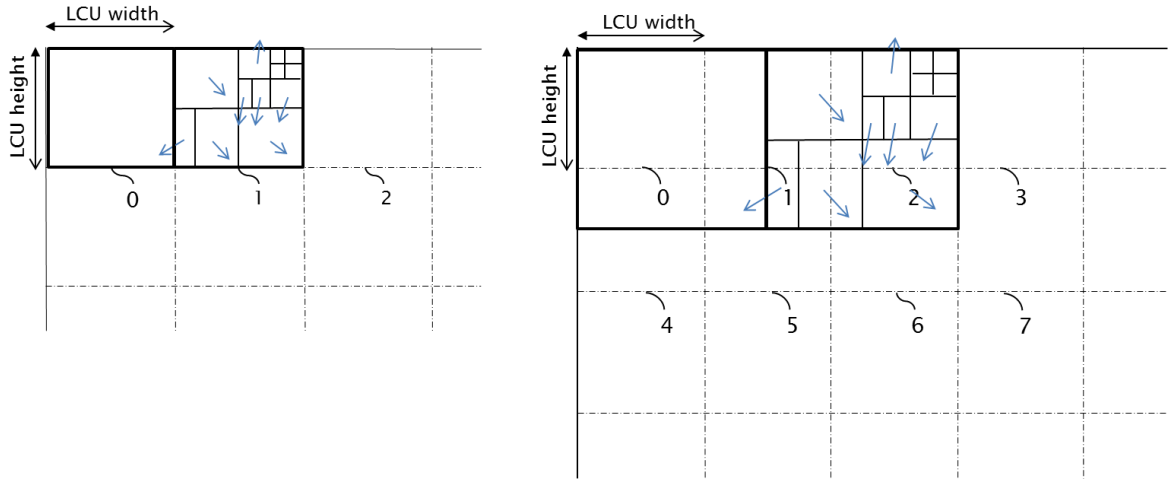

Figure 22: Up-scaled prediction information in case of ratio 1.5 based on the dyadic approach. This approach is not used.

The method adopted to derive prediction information in case of ratio 1.5 consists in the following, simpler, approach.

- First, each Largest Coding Unit (LCU) in the enhancement picture, the LCU is split into CUs with minimum size (4x4).
- Next, each so-obtained 4x4 is given the Prediction Unit type 2Nx2N.
- Finally, the prediction information of each obtained 4x4 Prediction Unit is computed as a function of prediction information associated to the co-located area in the base picture. The prediction information derived from the base picture include the following:
    - Prediction mode,
    - Merge information,
    - Intra prediction direction (if relevant),
    - Inter direction,
    - Cbf values,
    - Partitioning information,
    - CU size,
    - Motion vector prediction information,
    - Motion vector values (note the motion field is inherited before the motion compression that takes place in the base layer).
- Derived motion vector coordinates are computed as follows:

$$mv_x = mvbase_x \times \frac{PicWidthEnh}{PicWidthBase}$$

$$mv_y = mvbase_y \times \frac{PicHeightEnh}{PicHeightBase}$$

where (mv_x, mv_y) represents the derived motion vector, $(mvbase_x, mvbase_y)$ is the base motion vector, $(PicWidthEnh \times PicHeightEnh)$ and $(PicWidthBase \times PicHeightBase)$ are the respective sizes of the enhancement and base pictures.

    - Reference picture indices
    - QP value (used afterwards when applying the DBF onto the Base Mode prediction picture)

Therefore, each LCU of the enhancement picture is organized regardless the way the corresponding LCU in the base picture have been quad-tree represented.

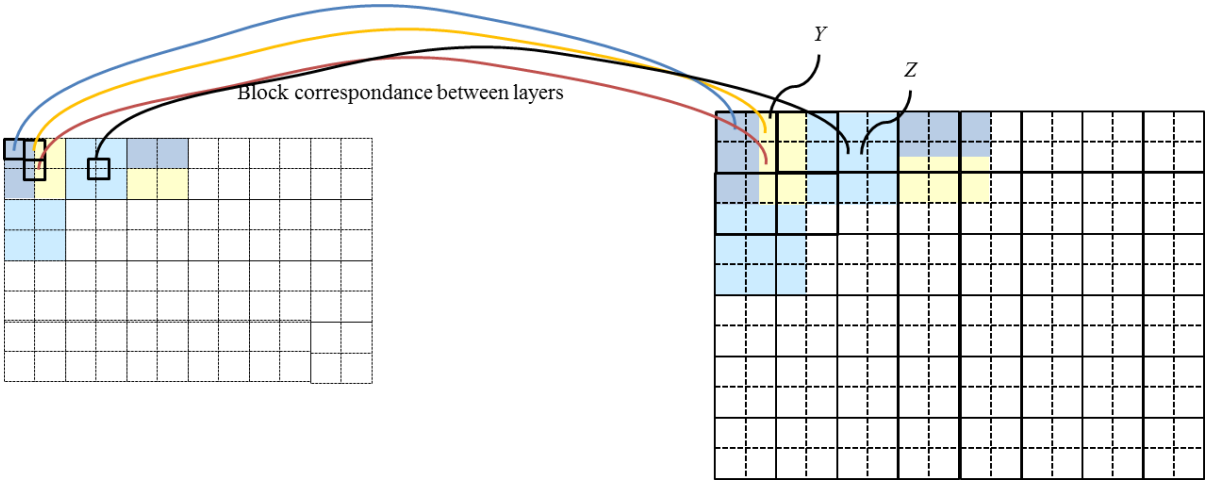

Figure 23: Inter-layer correspondence between elementary 4x4 blocks from the base and enhancement picture for ratio 1.5

The prediction information derivation presented above for ratio 1.5 aims at generating up-scaled prediction information that is used later during the predictive coding of motion information (see 2.4.1.4).
It can also be used in the construction of the Base Mode prediction picture. However, as already mentioned, the Base Mode prediction picture quality highly depends on the accuracy of the prediction information used for its construction.
Moreover, in case of scalability ratio 1.5, some constructed 4x4 Coding Units in the enhancement layer are fully confined inside their co-located 4x4 Coding Unit in the base picture, in terms of spatial localization. On the contrary, some other enhancement Coding Unit spatially overlaps several elementary 4x4 Coding Units in the base layer. This happens for enhancement CUs which coordinates (XCU, YCU) are such that:

$$(XCU\ mod\ 3 = 1)\ \ or\ (YCU\ mod\ 3 = 1)$$

For these particular Coding Units, we temporarily construct 2x2 Coding Units instead of 4x4 CUs. Therefore, each 2x2 CU contained in a 4x4 CU has a unique co-located CU in the base picture, hence inherits the prediction information coming from that co-located base CU. For example, the enhancement CU with coordinates (1, 1) on Figure 24 inherits prediction data from 4 different elementary 4x4 CUs in the base picture. Flow charts of Figure 25 and Figure 26 describe the overall prediction info derivation process.

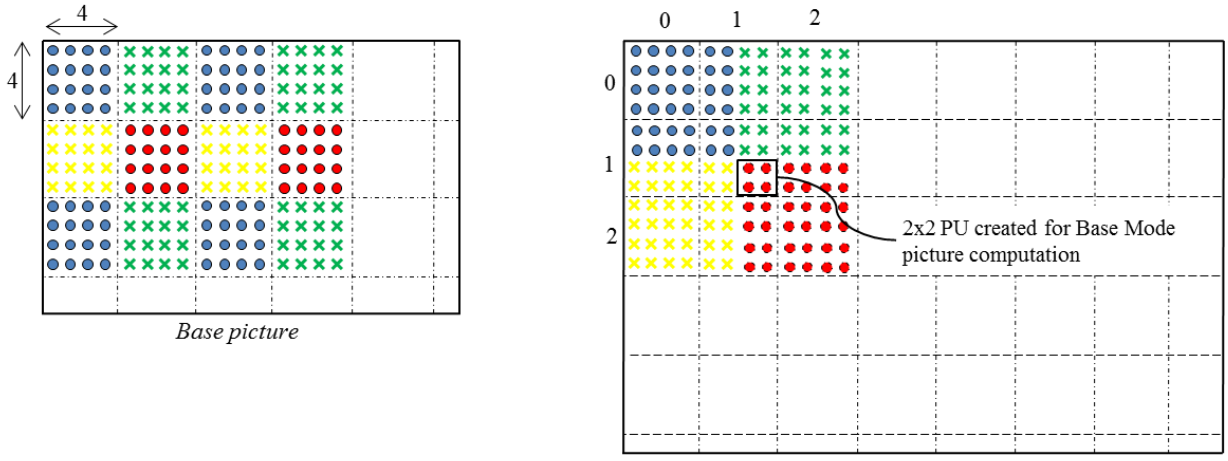

Figure 24: Derivation of 2x2 Prediction Units in case of spatial ratio 1.5

As a result of the prediction information up-sampling process designed for ratio 1.5 the Base Mode picture construction process is able to apply motion compensated temporal prediction on 2x2 CU's, hence benefits from all the prediction information issued from the base layer. The Base Mode prediction picture construction is described later in section 2.4.1.2.3.

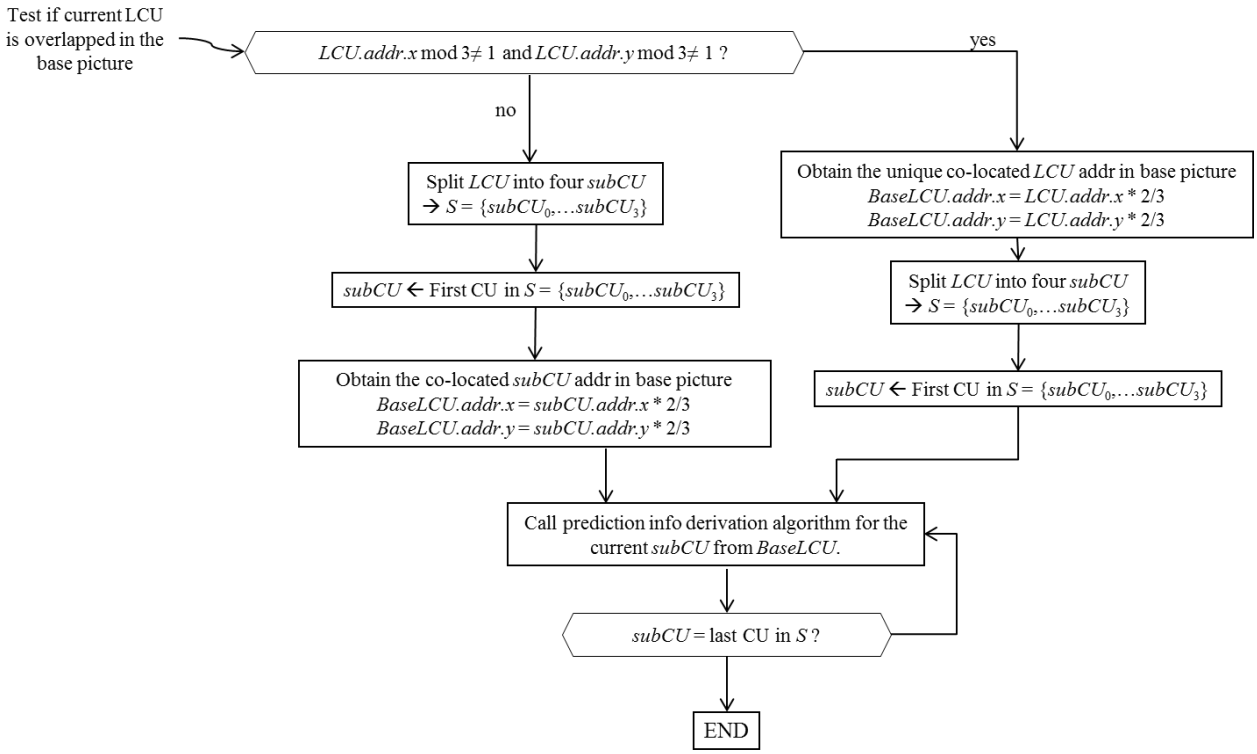

Figure 25: Overall prediction information derivation algorithm for spatial ratio 1.5

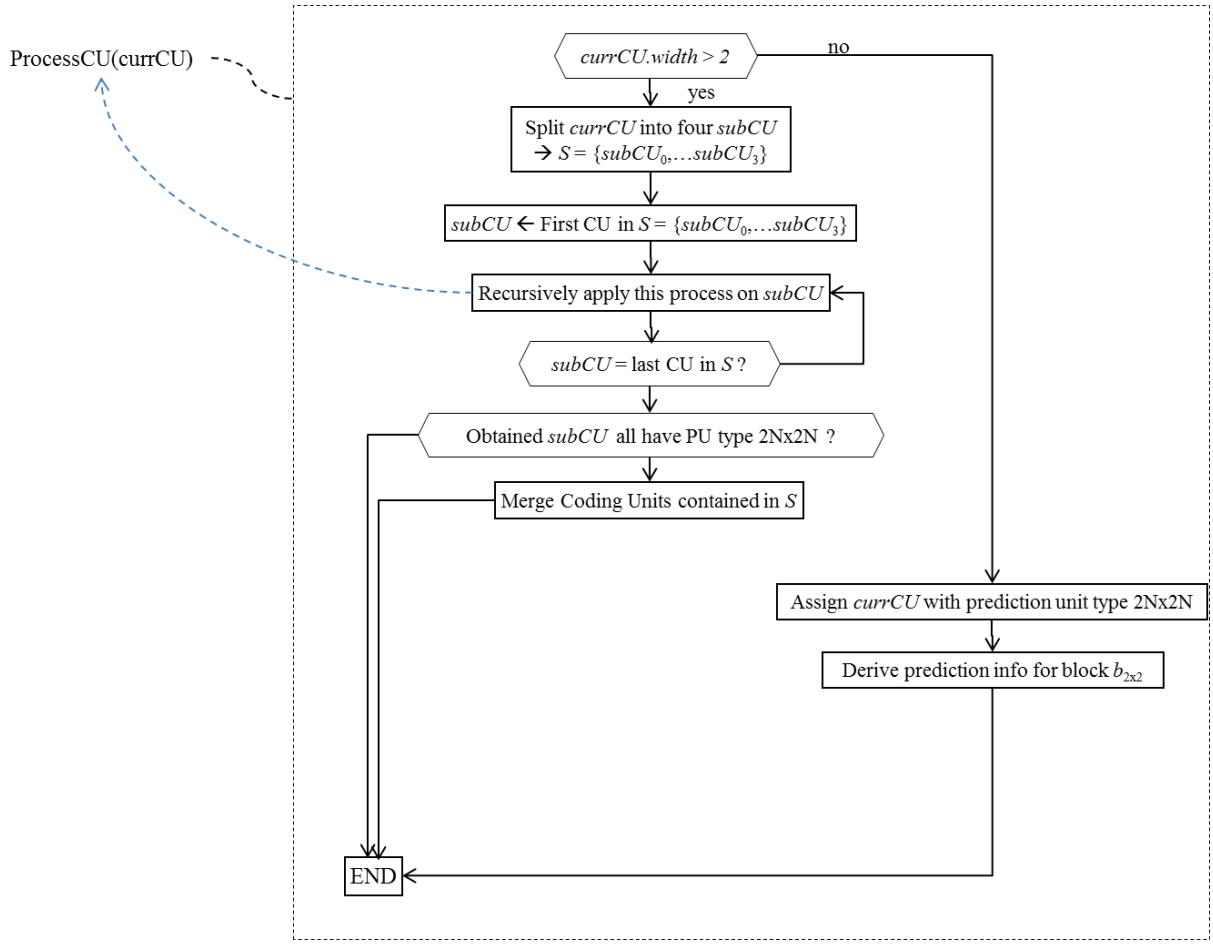

Figure 26: Prediction information derivation algorithm for current *subCU* from *baseLCU*

2.4.1.2.3 Inter-layer residual prediction for the Base Mode prediction picture computation

The temporal residual texture coded and decoded in the base layer is inherited from the base picture, and is employed in the Base Mode prediction picture computation. The Inter-layer residual prediction used here consists in applying a bi-linear interpolation filter on each Inter Prediction Unit contained in the base picture. This bi-linear interpolation of temporal residual is similar to the one that was used in H.264/SVC [3].

2.4.1.2.4 Construction of Base Mode prediction picture

Figure 27 depicts how the Base Mode prediction picture is computed. This picture is called "Base Mode", because it is predicted by means of the prediction information issued from the base layer. The inputs to this process are the following ones.

- The lists of reference pictures useful in the temporal prediction of current enhancement picture.
- The prediction information extracted from the base layer and re-sampled to the enhancement layer resolution. This corresponds to the prediction information resulting from the process of section 2.4.1.2.1.and 2.4.1.2.2
- The temporal residual data issued from the base layer decoding, and re-sampled to the enhancement layer resolution.
- The base layer reconstructed picture.

The Base Mode picture construction process consists in predicting each Coding Unit of the enhancement picture, conforming to the prediction modes and parameters inherited from the base layer.
It proceeds as follows.

- For each LCU in current enhancement picture
  - Obtain the up-sampled Coding Unit representation issued from the base layer
  - For each CU contained in current LCU
    - For each PU in current CU
      - Predict current PU with its prediction information inherited from the base layer

The PU prediction step proceeds as follows. In case the corresponding base PU was Intra-coded, then current PU is predicted by the reconstructed base PU, re-sampled to the enhancement layer resolution. In practice, the corresponding spatial area in the Intra BL prediction picture (see section 2.2) is copied. In case of an Inter coded base PU, then the corresponding PU in the enhancement layer is temporally predicted as well, by using the motion information inherited from the base layer. This means using the reference picture(s) in the enhancement layer that correspond to the same temporal position as the reference picture(s) of the base PU. A motion compensation step is applied by applying the motion vector inherited from the base onto these reference pictures. Finally, the up-sampled temporal residual data of the co-located base PU is applied onto the motion compensated enhancement PU, which provides the predicted PU in its final state.
Once this process has been applied on each PU in the enhancement picture, a full "Base Mode" prediction picture is available.

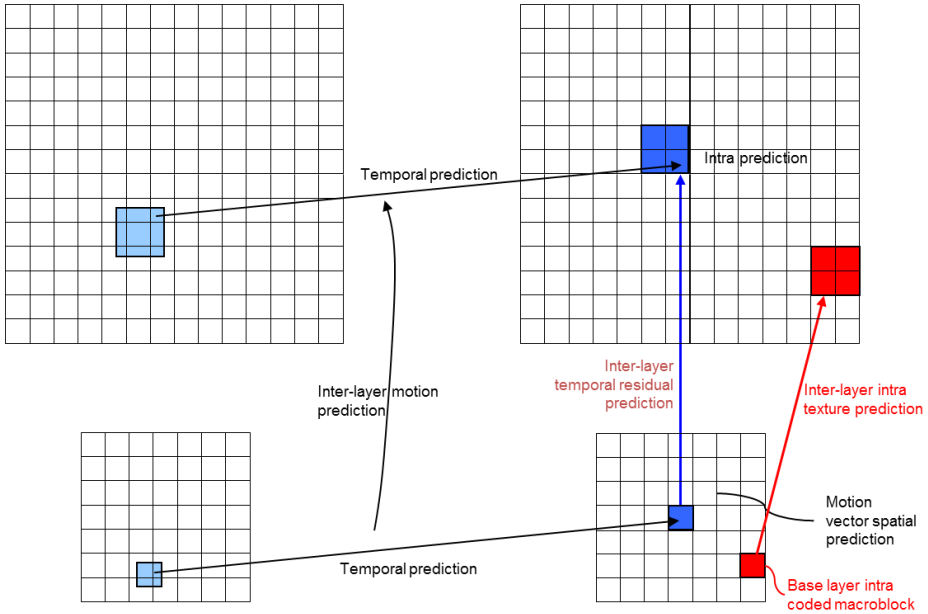

Figure 27: Temporal and spatial prediction to generate the Base Mode prediction picture

2.4.1.2.5 De-blocking filtering of Base Mode prediction picture

The last step in the Base Mode prediction picture computation consists in the de-blocking filter. To do so, each LCU of the enhancement layer is de-blocked by considering the Inter-layer derived CU structure associated to that LCU.
According to the default codec configuration, the de-blocking filter is applied on Coding Unit boundaries, and on Prediction Unit boundaries. Optionally the de-blocking can also be activated on Inter-layer derived Transform Units.
The Quantization Parameter (QP) used during the Base Mode picture de-blocking process is equal to the QP of the Co-located base CU of the CU currently being de-blocked. This QP value is obtained during the Inter-layer CU derivation step of section 2.4.1.2.1.
Finally, with respect to scalability ratio 1.5, the minimum CU considered during the de-blocking filtering step is 4x4. This means the de-blocking does not process 2x2 blocks frontiers inside 4x4 Coding Units, as illustrated on Figure 28.

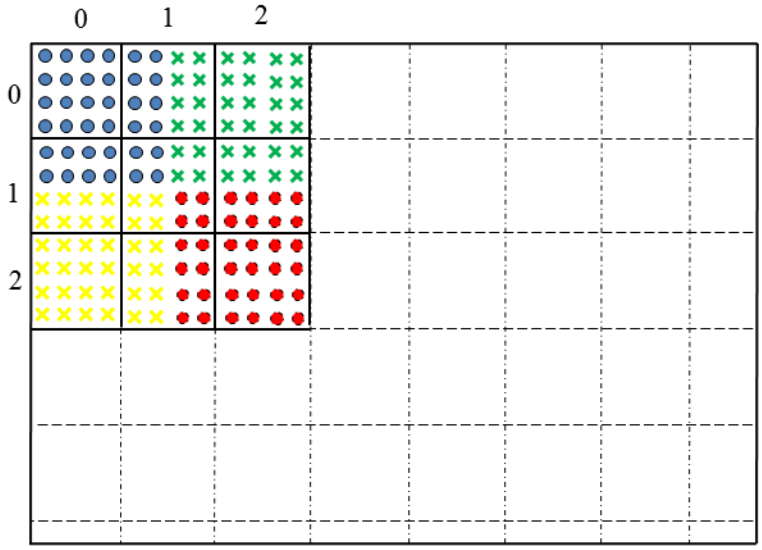

Figure 28: Considered boundaries processed during the Base Mode picture DBF (scalability ratio 1.5)

Once the Base Mode prediction picture is available in its deblocked version, then its pixel values can be used to serve as prediction data in the coding of enhancement Coding Units.

2.4.1.3 Generalized Residual Inter-Layer Prediction "GRILP"

This section introduced the so-called Generalized Residual Inter-Layer Prediction tool used in the proposed scalable HEVC codec. The former H.264/SVC [3] scalable video compression system employs Inter-layer prediction of temporal residual data. Inter-layer residual prediction is macroblock-based selected according to the rate distortion criteria.
In SVC, the base layer's residual data that serves as reference data for Inter-layer residual prediction corresponds to base Inter macroblocks residual in their decoded form. This was in-line with the single-loop decoding architecture chosen for the SVC standard.
Here, in the CSC codec, a multiple-loop decoding architecture is chosen. Therefore, more freedom is allowed in the codec design, since fully reconstructed base pictures are available. Moreover, one notices that in the former SVC approach, a decoded temporal residual block is associated with one motion vector. Then, in the temporal prediction process that takes place in the enhancement layer, if the enhancement motion vector is not correlated to the base CU's motion vector, then it is very likely that the residual associated to the base CU will not serve for Inter-layer residual prediction.
As a consequence, here we propose an improvement of SVC residual Inter-layer prediction which is adapted to a multiple-loop decoding approach. This tool is illustrated on Figure 29. It consists in generating, given a Prediction Unit and motion vector being considered, a corresponding temporal residual CU in the base layer.

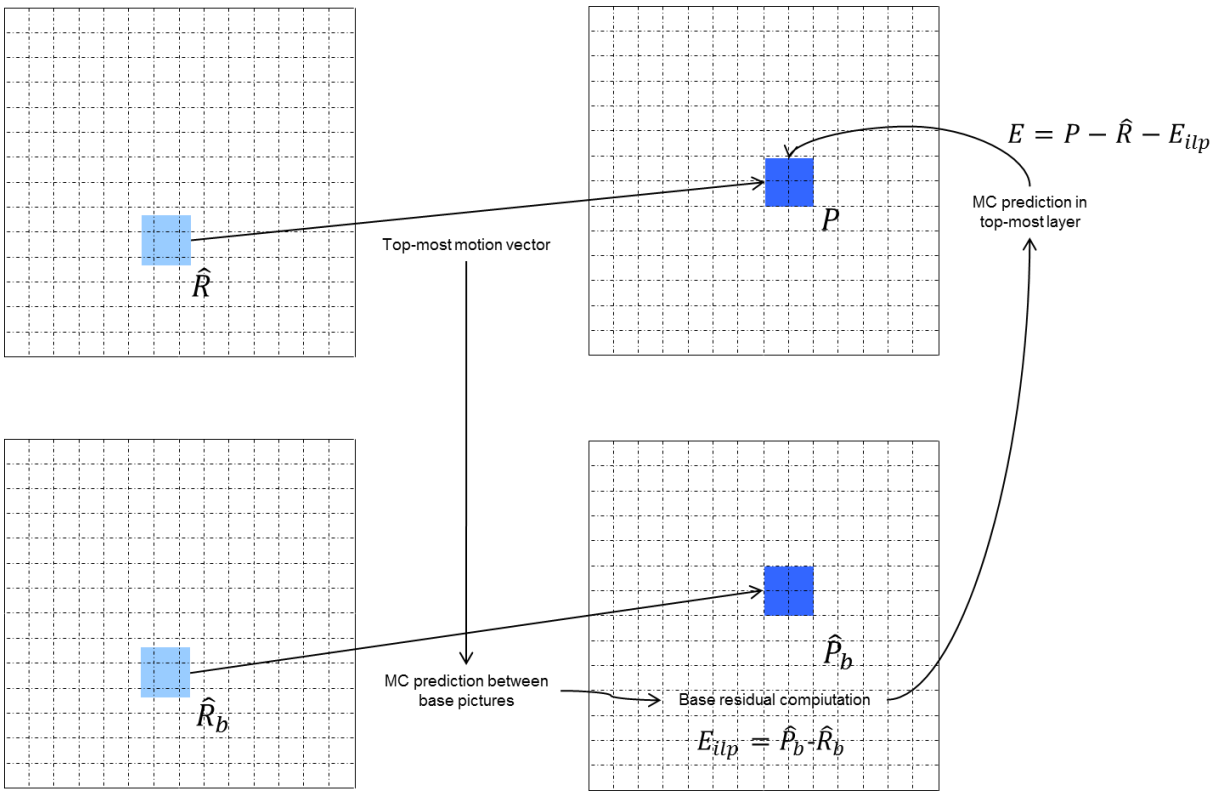

Figure 29: Generalized Residual Inter-Layer Prediction (encoder side)

The GRILP prediction mode is selected on a Coding Unit basis. It can be activated inside Inter Coding Units, hence is signaled in the bit-stream through a dedicated flag in the Coding Unit syntax for HEVC scalable extension (see section 3.5). Therefore, when activated, it is used for all Prediction Units (PU's) contained in the considered CU. The GRILP prediction mode proceeds as follows.

Reconstructed base layer pictures that are marked as reference for temporal prediction are stored in their up-sampled version on the encoder side. Then, during the motion search in the enhancement picture for a candidate Prediction Unit P in an original CU to encode, a number of candidate motion vectors are successively evaluated by computing their associated rate distortion cost. For each considered motion vector, corresponding to a candidate re-constructed reference block $\hat{R}$, a motion compensation step is applied between reconstructed base pictures in their up-sampled versions. This provides the new reference block $\hat{R}_b$ in the base picture of current reference picture. A new residual block is then computed in the base layer $E_{ilp} = \hat{P}_b - \hat{R}_b$, where $\hat{P}_b$ represents the co-located block of current enhancement Prediction Unit P being considered. This new base residual dynamically computed as a function of the enhancement motion vector is then used for Inter-layer prediction of the enhancement Coding Unit residual, which leads to the following residual data to encode in the enhancement layer:

$$E = P - \hat{R} - E_{ilp}$$

On the decoder side, the same prediction process as on the encoder side can be performed. This is what is done in the current decoder implementation. However, some significant amount of memory can be saved by proceeding the following way, as illustrated on Figure 30. Reconstructed base pictures that serve for temporal prediction of subsequent pictures are no more stored in their up-sampled version. On the opposite, when a Coding Unit uses the GRILP prediction mode, then the spatial area $\hat{r}_b$ from the base picture corresponding to the concerned enhancement PU is being up-sampled, according to the texture up-sampling process of section 2.2, which provides the up-sampled block $\hat{R}_b$, which is identical to the block $\hat{R}_b$ defined previously. It is then used in order to perform a motion compensated temporal prediction of the co-located base area $\hat{P}_b$ of current Prediction Unit P. Such prediction leads to the base residual block $E_{ilp} = \hat{P}_b - \hat{R}_b$. The remaining of the GRILP process is then identical to that of the encoder side.

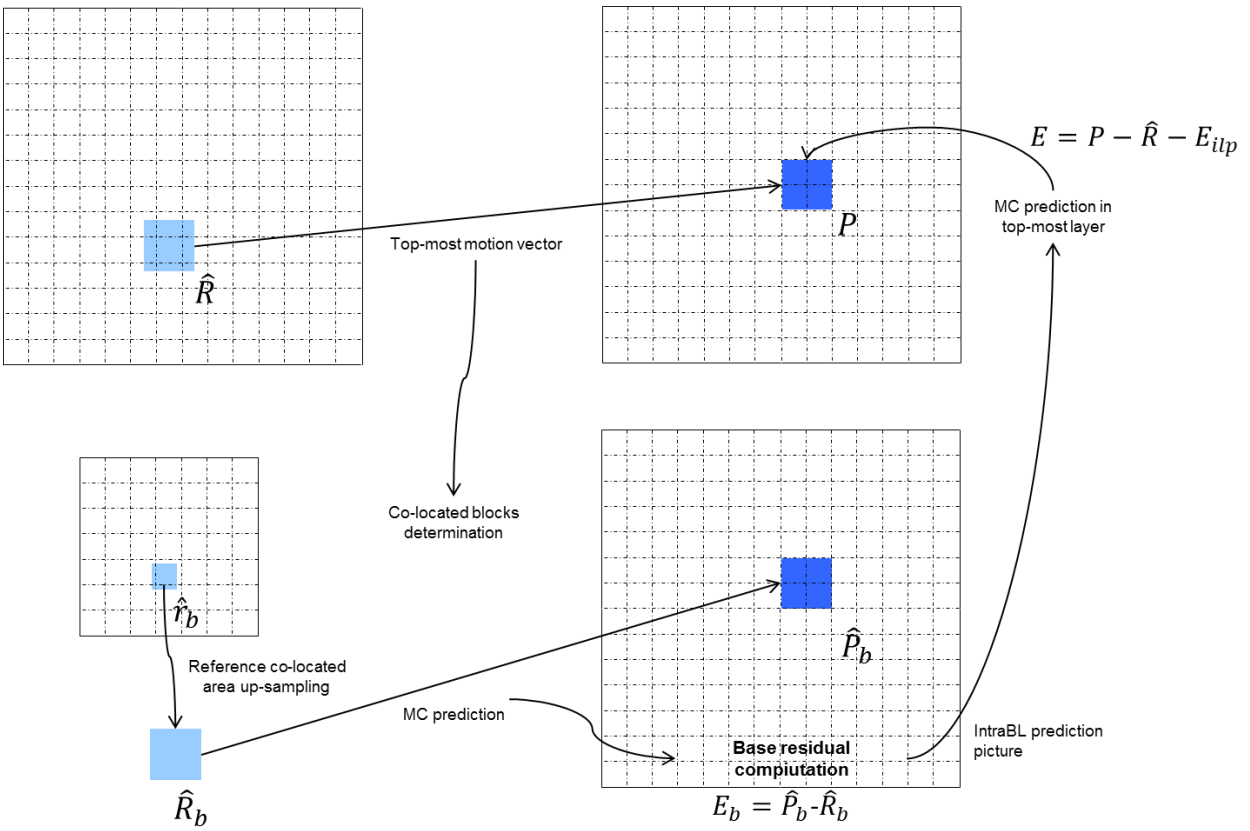

Figure 30: Generalized Residual Inter-Layer Prediction (decoder side)

2.4.1.4 Coding of prediction information in the enhancement layer

This section describes the tools that are introduced in CSC proposal, in the field of prediction information coding in the enhancement layer. The tools described in this section only concern the coding/decoding of Inter enhancement pictures. The Merge derivation predictors and the AMVP process specified for HEVC are adapted to the enhancement layer prediction in order to improve the coding efficiency of the Inter, Merge, Merge Skip and GRILP modes.

2.4.1.4.1 AMVP derivation in enhancement Inter pictures

As for the Merge mode candidate derivation, the number of predictors and the order of predictors are modified and a new predictor is added for AMVP in the enhancement layer compared to HM6.1 as described in the following:

- The number of predictors in the final list is increased from 2 to 3.
- The temporal is the first candidate of the list instead of the last one for the base layer. Moreover, only the centered position of the temporal predictor is used.
- The availability check of the spatial predictor is modified in order to select the motion vectors of neighboring blocks which were encoded with the base mode (see section 2.4.1.4.2). The base layer motion vector located at the center of considered enhancement CU is added to the motion prediction list.
- The co-located motion vector in the base picture, re-scaled to the resolution of the enhancement layer, is added to the list of predictors. This mechanism proceeds the same way as in [4]. The base layer motion vector that is used here is located at the center of considered enhancement CU.
- No specific syntax for Inter-layer prediction of motion vector is needed.

2.4.1.4.2 Spatial prediction of motion data in the enhancement picture

Spatial prediction of motion data in HEVC has been modified in the CSC proposal. It concerns the specific Base Mode prediction mode, and its use in AMVP and Merge processes..
Indeed, the Base Mode prediction mode is not coded as a particular Inter coding mode, as it will be described in section 3.5. On the contrary, the Base Mode and Intra BL coding modes are coded as so-called “Inter-Layer” prediction modes in the proposed Coding Unit syntax.

Moreover, BaseMode Coding Units contain some relevant temporal prediction information, in case at least one of its underlying CU(‘s) was Inter coded in the base layer. Therefore, it has been chosen to enrich the spatial derivation of temporal prediction information of HEVC for the enhancement Inter pictures. The modified process includes temporal prediction information associated to Base Mode Coding Units where possible.

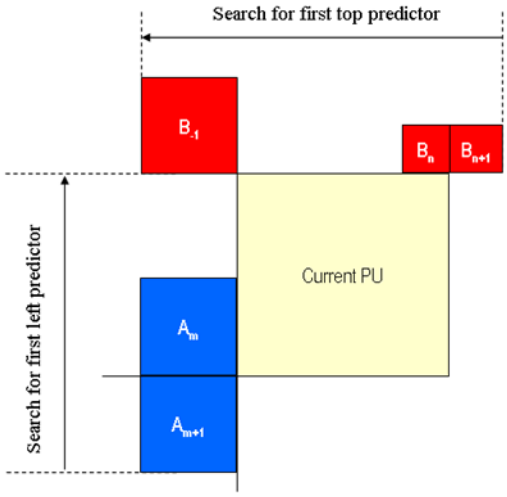

Figure 31: Spatial derivation of motion data in HEVC

As a result, when filling the sets of AMVP and optionally MERGE spatial predictors, the test done on the coding mode of the spatial neighboring CU’s has been modified. In HEVC, this test consists in enabling the derivation of only Inter neighboring Prediction Units.
The modified approach consists in also enabling the spatial derivation not only of Inter neighboring CU’s, but also Base Mode neighboring Coding Units.

NOTE: In the CSC source code, the spatial derivation of the Base Mode Prediction Unit is only allowed during the Advance Motion Vector Prediction process. An alternative approach may consist in also allowing it during the Coding Unit MERGE mechanism of HEVC. This can be activated through a dedicated option in the CSC source code.

2.4.1.4.3 Merge derivation

Figure 32 shows the flow chart of the derivation process of the Merge, Merge Skip and Merge GRILP Modes. Compared to the HM6.1, the number of candidates and the order of candidates are modified. Moreover, new candidates are added for the enhancement layer. These adaptations are listed in the following:

- The number of candidates in the final list is increased from 6 to 10.
- The temporal candidate is now the first candidate in the list. Compared to the temporal Merge candidate of the base layer, only the centered position is considered for the enhancement layer. Moreover, the memory compression of motion vectors is disabled for the enhancement layer.
- The spatial candidates are added in the list after the temporal candidate. The derivation process of spatial candidates is the same as for the base layer. In addition, if the neighboring block is encoded with the base mode this candidate is not added in the Merge list.
- Then the motion vector of the co-located base layer is added to the list of candidates.
- The base motion vector is followed by 4 offset candidates derived according to this base motion vector candidate. The generation of these candidates consists in adding one offset (offset value: 4 or -4) alternatively to the horizontal or to the vertical component of the motion vector from list L0 and its inverse value to components of motion vector from list L1.
- A duplicate check is used on this list of maximum 10 candidates. Then the combined candidates derivation and the zero candidates derivation are applied if at least one position is available.

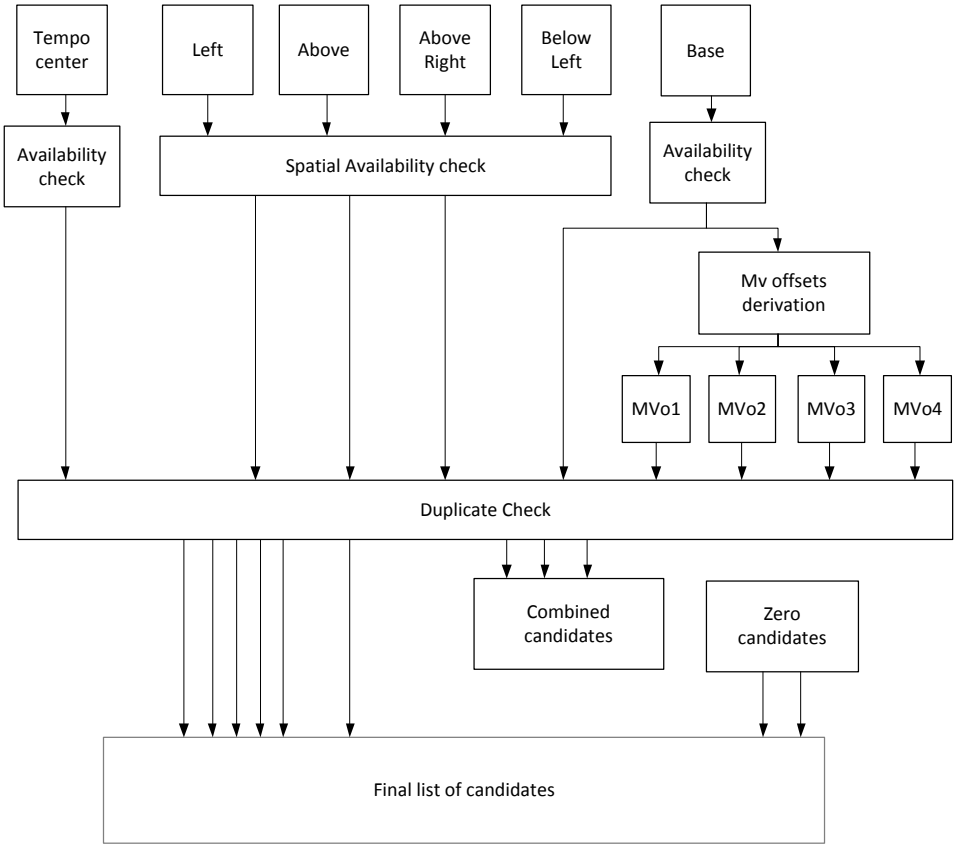

Figure 32 Merge, Skip Merge and Merge GRILP motion vector predictor derivation

2.4.2 Encoder control methods used in enhancement Inter sequences

This section describes the encoder control methods used in the proposed scalable HEVC codec. First, picture level Lagrange parameter and QP assignment process is presented (section 2.4.2.1).

2.4.2.1 Picture level coding parameters assignment

Figure 33 illustrates the hierarchical B picture coding structure used in the random access testing conditions in proposed codec. Quantization parameters assigned to each picture are also shown. The coding structure and QP configurations employed in the base layer are strictly identical to those used in the HM6.1 random access testing conditions. Therefore, the coded base layers are strictly the same as the base layer anchors generated to the scalable HEVC call for proposal **[2**].

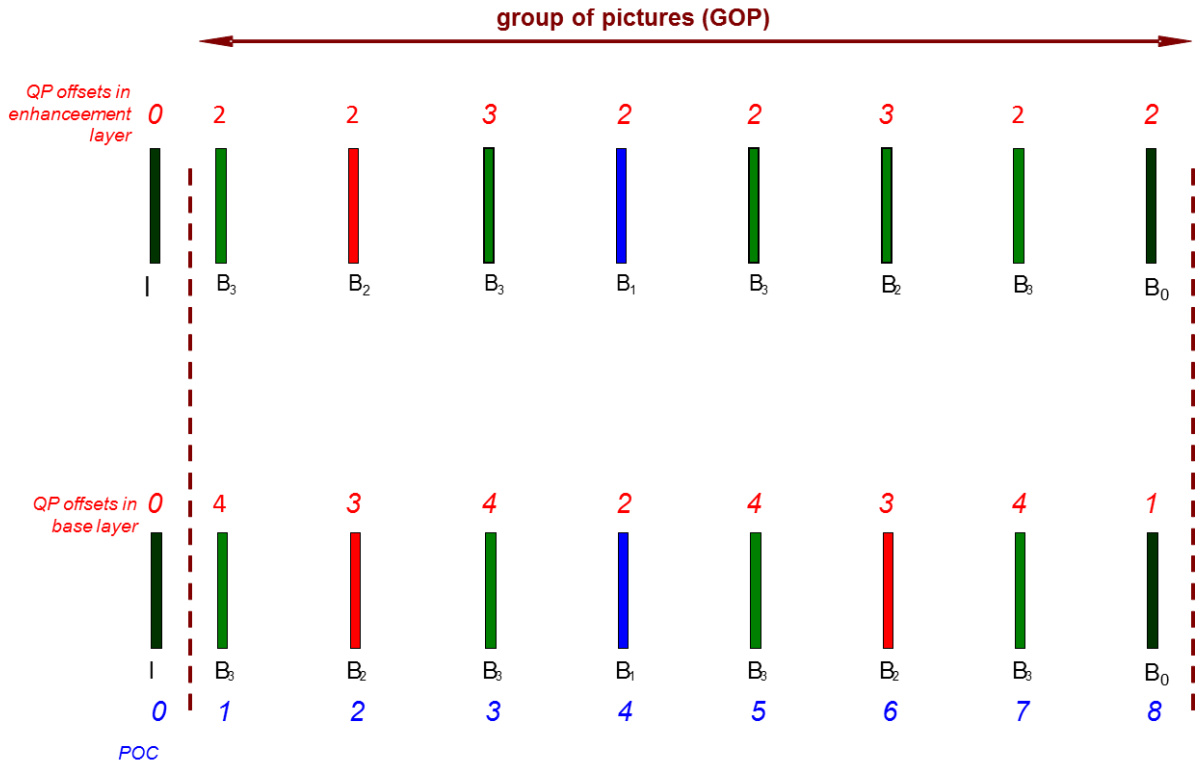

Figure 33: QP settings in Base and Enhancement layers

As shown by Figure 33, the coding structure used in the enhancement layer is identical to that of the base layer. On the contrary, the QP's used for the enhancement B pictures are different. They conform to the chosen rate allocation between hierarchical B pictures, illustrated on the diagram of Figure 34. The shown QP's are added to the global sequence level QP in order to compute a picture level QP value. The QP offset assignment of Figure 34 is employed respectively in the base layer (blue line) and in the enhancement layer (red line). These QP values are used to compute the rate distortion slope used afterwards in the rate distortion optimization process.

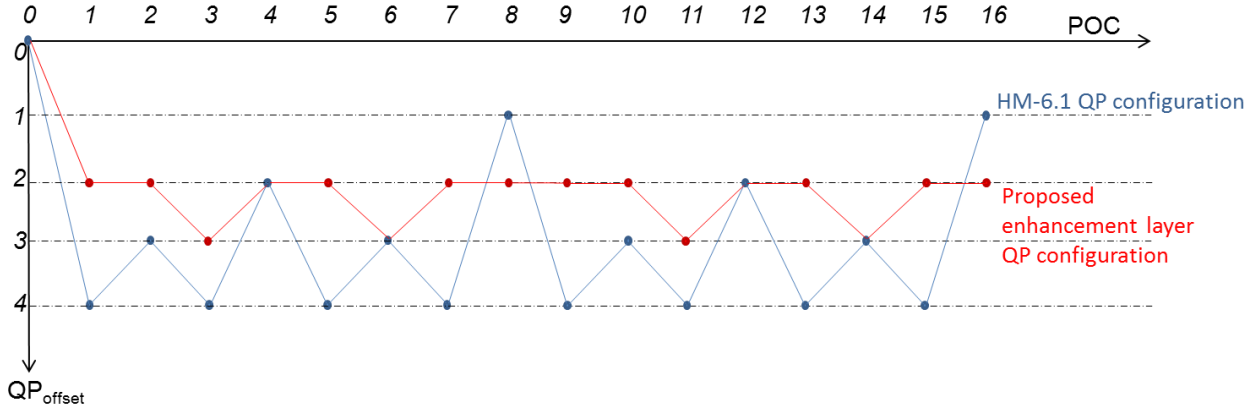

Figure 34: Rate allocation used between hierarchical B pictures in enhancement layer. POC 0 corresponds to an IDR picture, and other POC to hierarchical B pictures.

It can be noticed that the proposed bitrate allocation tends to balance the bitrate between pictures, compared to initial HM6.1 configuration.

Given the QP offset configuration chosen to encode enhancement pictures, the encoder computes the quantization parameter and Lagrange parameter used hereafter to encode enhancement slices. This process is illustrated on Figure 35. Once the QP offset for a considered enhancement picture is obtained, then a QP value for that picture is computed $QP = QP_{video} + QP_{offset}$, where the QP_{video} parameter is calculated as a function of an input λ_{video} parameter: $QP_{video} = 3log_2(\lambda_{video}[layerId]) + 12$. Then a Lagrange parameter λ_{final} is computed as illustrated by Figure 35. This calculation is taken as is from the HM6.1.

The next step depends on the type of enhancement picture being encoded.

In case of an Inter enhancement picture, the quantization parameter that is finally used during the actual quantization process in Inter is computed: $QP_{final} = 4.2005 \times \ln(\lambda) + 13.7122$. This relationship between the QP and λ is taken from [11].
NOTE: according to an option in the proposed codec, this last QP value computation can be de-activated. In that case, the quantization parameter used during quantization is equal to $QP_{video} + QP_{offset}$.

The Intra picture case is handled as follows. As previously stated, Intra enhancement pictures of Inter sequences are coded and decoded according to the Intra texture coding process described in section 2.3. However, the balanced encoding between frames and components introduced in section 2.3.7 is modified, so as to properly balance the coding between Intra and Inter pictures.
To do so, the luminance frame merit $M_{F,Y}$ is now calculated as a function of the Lagrange parameter associated to the considered Intra enhancement slice, according to the following equation:

$$M_{F,Y} = \lambda_{I\_scale} \times \lambda_{final}$$

Where λ_{I_scale} represents a scaling factor used to balance the coding between Intra and Inter enhancement pictures. The chrominance frame merits $M_{F,U}$ and $M_{F,V}$ are then computed according to the luma/chroma balancing method described in section 2.3.7.2.

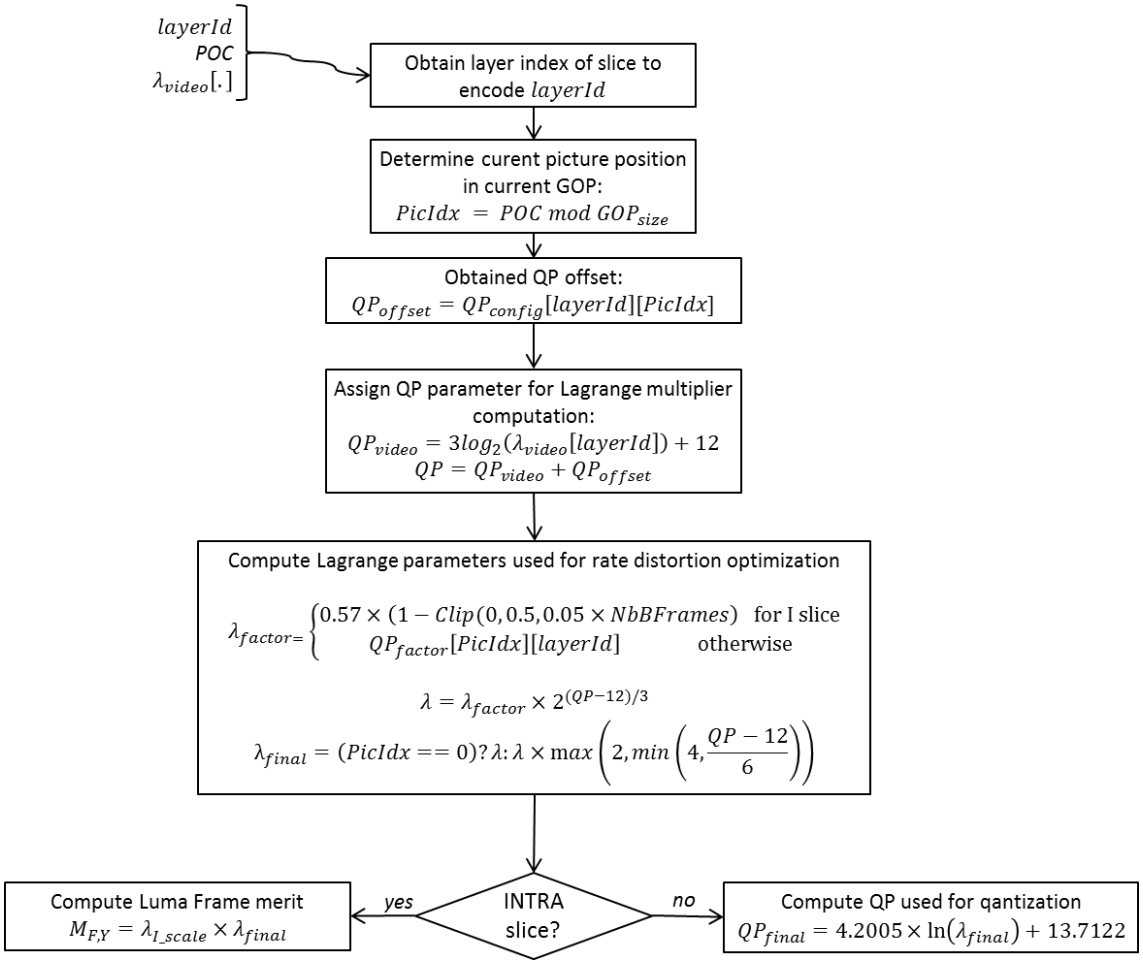

Figure 35: Computation of Lagrange parameter and QP used to encode an enhancement slice

2.4.2.2 Slice-level dynamic coding modes activation in Inter enhancement slices

The modes usage may significantly and dynamically vary from one slice to another, depending on the picture content, the temporal layer, the QP. For instance at high QP's skip mode is the most used mode, while at medium QP's, Inter-layer modes are very useful. To reduce the encoder computation and to save useless signaling bits, an adaptive modes activation process is used. The principle is to activate the modes for a given slice based on statistics measured in a previous slice close temporally and of same of previous temporal layer.
When processing each Inter slice, statistics related to their encoding modes are computed. They consist for each enabled mode m in:

- The ratio $ratio_{CUs}$[m] between the number of CU's using this mode and the total number of CU's;

- The ratio $ratio_{Pixels}[m]$ between the number of pixels using this mode and the total number of pixels (that is, the slice size in pixels).

When processing a given picture, first its 'dependency' picture is identified. This picture is the closest previously coded picture of the GOP of same type as the current picture and:

- of same temporal layer if such a picture exists
  or
- from the immediate lower temporal layer

The dependency structure is illustrated in Figure 36 in the case of a GOP size of 8. The dependency is not allowed across GOPs. For the first B slice of each GOP, all modes are made enabled.
A mode m of the current picture is activated if it was already active in its dependency picture and if the following condition is true:

$$\text{Min}(ratio_{CUs}[m] , ratio_{Pixels}[m]) \geq Th$$

where Th is a given threshold (practically set to 0.01). If no dependency picture is found, all modes are enabled for the current picture.
Activated/de-activated modes are signaled in the slice header. This relates to the following modes:

- Intra mode
- Inter-layer modes 'Base mode' and 'Intra BL' (activated or de-activated together)
- GRILP mode

Skip and Inter modes are always activated.

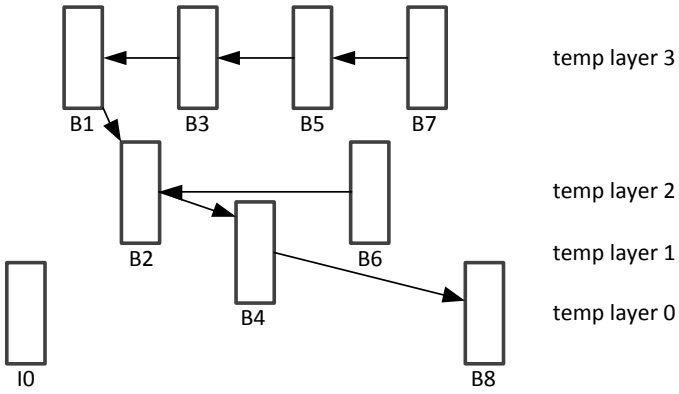

Figure 36: Dependency structure for adaptive modes activation.

2.4.2.3 Rate distortion optimization performed on the Coding Unit level

The encoder control process used in Inter enhancement picture includes the well-known mode decision process, which searches the best coding representation of each Largest Coding Unit (LCU) in a considered enhancement picture.
This involves a recursive search of the best quad-tree representation of each Largest Coding Unit. At each possible quad-tree decomposition depth, a set of Coding Units is considered. For each of these Coding Units, a set P of possible coding modes is being considered. The coding mode that minimizes the Lagrangian cost function if selected for considered CU:

$$p_{opt} = Argmin_{p \in P}\{D(p) + \lambda . R(p)\}$$

where the parameter λ is equal to the λ_{final} value previously obtained for Luma component, and weighted for chroma component in the same way as in the HM6.1 coding process [12].
Once the best CU coding configuration is obtained at each LCU quad-tree depth level, then the best quad-tree representation is determined for that LCU. This is done, during the recursive LCU processing, by performing a bottom-to-top rate distortion cost minimization [12].
Therefore, the coding mode rate distortion optimization process is quite similar to that of HM6.1, except that for each considered Coding Unit the set of HEVC coding modes P is extended with all Inter-layer prediction tools presented in section 2.4.1.

2.5 In-loop filtering

2.5.1 De-blocking filter

Section 2.3.1.1 described several block types used in IDR slices, which were characterized by their sizes and labels. The structure for a frame made out of those blocks can be translated to classical HEVC depth and transform quad-trees. As a consequence, in IDR slices, classical HEVC deblocking is applied to the blocks of that structure, considering the blocks as intra with their coded block flags set to 0, and having a QP for all components derived as described in section 2.3.11.
Other frames, following a classical HEVC encoding, go through the normal HEVC deblocking filter.

2.5.2 SAO (Sample Adaptive Offset)

The SAO process for the enhancement layer is different from the SAO process of the HEVC base layer. In the enhancement layer, the process is picture based instead of LCU based in the base layer. Moreover, some modifications of the implementation of the HM6.1 SAO picture based are used to improve the coding efficiency. These modifications have been presented in JCTVC-G490 [13] and JCTVC-G246 [14] and give coding efficiency improvements. The adaptation of JCTVC-G490 for enhancement layer consists in using 4 or 8 offsets corresponding to the 4 classical classifications or 4 additional classifications. In this classification, the sign of offsets for Edges offset is encoded. The adaptation of JCTVC-G246 for enhancement layer consists in adding 8 SAO Band offset classifications. With this modification the 9 SAO band offset classifications are the combination of 3 interval sizes and 3 sizes of class.

2.5.3 Adaptive Loop filter

The Adaptive Loop Filtering tool of HEVC is employed in the enhancement layer coding in the proposed scalable codec. In Inter pictures, the HEVC Adaptive Loop Filtering process is used as is. This correspond to the picture-based syntax as found in HM6.1 and described in JCTVC-H0274 [15], using the single filter type (9x7 cross + 3x3 square) as described in JCTVC-H0068 [16]. In Intra pictures, the HEVC ALF is also used as is, except that the Lagrange parameter used for ALF is computed according to the Intra frame Merit, in the same way as for SAO (see section 2.5.2)

2.6 Internal bit-depth used in the enhancement layer

In both the All-Intra and Random Access configurations, the enhancement pictures are processed with 10-bit internal bit-depth, while the base layer processing is kept with 8-bit internal bit-depth, conforming to HM6.1[12].
To do so, some inter-layer prediction data is being scaled to provide the enhancement layer coding/decoding process with 10-bit prediction data. This up-scaling includes the following.
The base picture samples are scaled to 10-bit values before undergoing the spatial up-sampling process of section 2.2. Once the spatial up-sampling is done, the base picture is restored in 8-bit representation, for the coding/decoding of subsequent base pictures. When the GRILP coding mode is active, the up-sampled base pictures used to compute dynamic temporal residual blocks (see section 2.4.1.3) are stored in memory with 10-bit precision.
The temporal residual data issued from the decoding of the base picture are up-scaled before starting to use them in the computation of the Base Mode prediction picture (see section 2.4.1.2).

3 Syntax and semantics description

This section provides the additional syntax elements and their associated semantics brought to the HEVC specification, in the proposed CSC scalable codec.

3.1 New NAL unit types for the enhancement layer

The CSC proposal defines two new NAL units to introduce the coded data of the enhancement layers. These two NAL units corresponds to the "**nal_unit_type"** 20 and 21 as presented in the table.

nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0	Unspecified	non-VCL
1	Coded slice of a non-IDR, non-CRA and non-TLA picture slice_layer_rbsp( )	VCL
2	Reserved	n/a
3	Coded slice of a TLA picture slice_layer_rbsp( )	VCL
4	Coded slice of a CRA picture slice_layer_rbsp( )	VCL
5	Coded slice of an IDR picture slice_layer_rbsp( )	VCL
6	Supplemental enhancement information (SEI) sei_rbsp( )	non-VCL
7	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
8	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
9	Access unit delimiter access_unit_delimiter_rbsp( )	non-VCL
10-11	Reserved	n/a
12	Filler data filler_data_rbsp( )	non-VCL
13	Reserved	n/a
14	Adaptation parameter set aps_rbsp( )	non-VCL
15-19	Reserved	n/a
20	**Coded slice of an IDR picture in scalable extension**	**VCL**
21	**Coded slice of an non-IDR picture in scalable extension**	**VCL**
22-23	Reserved	n/a
24..63	Unspecified	non-VCL

In the text, coded slice NAL unit collectively refers to a coded slice of a non-IDR picture NAL unit or to a coded slice of an IDR picture NAL unit. The variable IdrPicFlag is specified as

IdrPicFlag = (((nal_unit_type = = 5) || (nal_unit_type = = 20)) ? 1 : 0)

3.2 Modification of the NAL unit

nal_unit(NumBytesInNALunit) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
NumBytesInRBSP = 0	
temporal_id	u(3)
~~**reserved_one_5bits**~~	~~u(5)~~
layer_id	u(5)
for(i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) = = 0x000003) {	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
}	
}	

3.3 Adaptation parameter set RBSP syntax (which includes the probability parameters for the Intra picture)

The syntax regarding the APS NAL unit has been modified as follows to include the different statistical parameters regarding the Intra picture as described in section 2.3.10.

aps_rbsp( ) {	**Descriptor**
aps_id	ue(v)
aps_scaling_list_data_present_flag	u(1)
if(aps_scaling_list_data_present_flag)	
scaling_list_param( )	
aps_deblocking_filter_flag	u(1)
if(aps_deblocking_filter_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
aps_sao_interleaving_flag	u(1)
if(!aps_sao_interleaving_flag) {	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag)	
aps_sao_param( )	
}	
aps_adaptive_loop_filter_flag	u(1)
if(aps_adaptive_loop_filter_flag)	
alf_param( )	
aps_extension_flag	u(1)
if(aps_extension_flag)	
aps_residual_param( )	
while(more_rbsp_data( ))	
aps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

aps_residual_param( ) {	**Descriptor**
for(t = 0; t< number_type_32; t++) {	
block_type_32[t]	u(1)
texture_32 |= **block_type_32**[t]	
}	
for(t = 0; t< number_type_16; t++) {	
block_type_16[t]	u(1)
texture_16 |= **block_type_16**[t]	
}	
for(t = 0; t< number_type_8; t++) {	
block_type_8[t]	u(1)
texture_8 |= **block_type_8**[t]	
}	
re_stat_flag	u(1)
if (re_stat_flag == true) {	
for(b = 0; i<number_proba; b++) {	
for(t = 1; t< number_type_32; t++) {	
if (CU_type_32[t] == 1) {	
block_proba_32[b*(number_type_32+1)+t]	u(8)
}	
if (texture_32 ==1) {	
block_proba_32[b*(number_type_32+1)]	u(8)
}	
}	
}	
for(b = 0; i<number_proba; b++) {	
for(t = 1; t< number_type_16; t++) {	
if (block_type_16[t] == 1) {	
block_proba_16[b*(number_type_16+1)+t]	u(8)
}	
if (texture_16 ==1) {	
block_proba_16[b*(number_type_16+1)]	u(8)
}	
}	
	u(1)
for(b = 0; i<number_proba; b++) {	
for(t = 1; t< number_type_32; t++) {	
if (block_type_8[t] == 1) {	
block_proba_8[b*(number_type_8+1)+t]	u(8)
}	
if (texture_8 ==1) {	
block_proba_8[b*(number_type_8+1)]	u(8)
}	
}	

}	
}	
for(t = 1; t< number_type_32; t++) { /* For Block type 32x32 */	
aps_residual_stats(10) /* For Y component */	
aps_residual_stats(8) /* For U component */	
aps_residual_stats(8) /* For V component */	
}	
for(t = 1; t< number_type_16; t++) { /* For Block type 16x16 */	
aps_residual_stats(8) /* For Y component */	
aps_residual_stats(6) /* For U component */	
aps_residual_stats(6) /* For V component */	
}	
for(t = 1; t< number_type_8; t++) { /* For Block type 8x8*/	
aps_residual_stats(6) /* For Y component */	
aps_residual_stats(4) /* For U component */	
aps_residual_stats(4) /* For V component */	
}	
}	

Where number_type_32 =10, number_type_16 =10 and number_type_8 =19 in the current CSC proposal.

aps_residual_stats (n) { /*Statistics for each CU type */	**Descriptor**
stat_presence_flag	u(1)
if (stat_presence_flag == 1) {	
num_coeff_minus_1	u(n)
num_bit_minus_1	u(4)
m = num_bit_minus_1+1;	
for (t = 0; t< 2^n; t++) {	
while (num_coeff_minus_1!= 0)	
data_flag	u(1)
if (data_flag == 1) {	
alpha	u(3)
beta	u(m)
}	
}	
}	
}	
}	

3.4 Slice-level syntax

The slice header syntax for the enhancement pictures is modified compared to the slice header syntax in the HM6.1 specifications. Some flags are inserted to signal the use of various scalable coding modes in the considered enhancement slice.
These flags are the following ones:

- **enable_grilp_flag**: indicates that the GRILP prediction mode is enabled in the current slice of the enhancement layer.
- **enable_intra_pred_flag**: indicates that the use of the classical HEVC intra prediction is enabled in the current slice of the enhancement layer.
- **enable_inter_layer_flag**: indicates that the use of the inter-layer prediction modes "IntraBL" and "Base Mode" are enabled in the slice of the enhancement layer. As presented in section 2.4.1.4, the syntax element **six_minus_max_num_merge_** specifies the maximum number of merging MVP candidates supported in the slice for the enhancement layers.

slice_header( ) {	**Descriptor**
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag && nal_unit_type != 20) {	
pic_parameter_set_id	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(**separate_colour_plane_flag** = = 1)	
colour_plane_id	u(2)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
} else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt[i]	ue(v)
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt_minus1[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)
if(slice_sao_interleaving_flag && slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
}	
if(scaling_list_enable_flag || deblocking_filter_in_aps_enabled_flag || (sample_adaptive_offset_enabled_flag && !slice_sao_interleaving_flag) || adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
if(slice_type = = P || slice_type = = B) {	

num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(lists_modification_present_flag) {	
ref_pic_list_modification( )	
ref_pic_list_combination( )	
}	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
}	
if(cabac_init_present_flag && slice_type != I)	
cabac_init_flag	u(1)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_in_aps_enabled_flag)	
inherit_dbl_params_from_aps_flag	u(1)
if(!inherit_dbl_params_from_aps_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
}	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(slice_type != I && ((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_idc = = 1 && slice_type = = B))	
pred_weight_table( )	
}	
if(slice_type = = P || slice_type = = B)	
if (layer_id > 0) {	
six_minus_max_num_merge_cand	ue(v)
} else {	
five_minus_max_num_merge_cand	ue(v)
}	
enable_grilp_flag	u(1)
enable_intra_pred_flag	u(1)

enable_inter_layer_flag	u(1)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(tiles_or_entropy_coding_sync_idc > 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

3.5 Coding Unit (CU) and Prediction Unit (PU) syntax

The syntax element of the Coding Unit as well as the Prediction Unit have been modified to introduce an interlayer flag called **inter_layer_flag** to inherit some information from the base layer. The **inter_layer_mode** syntax element enables to select one of the two modes between "IntraBL" and "BaseMode".

- **inter_layer_flag**: indicates if the Inter-layer prediction modes "IntraBL" or "Base Mode" are used in the considered Coding Unit.
- **inter_layer_mode:** indicates which mode is used for the considered Prediction Unit. 0 means "IntraBL" and 1 means "Base Mode".

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
CurrCbAddrTS = MinCbAddrZS[x0 >> Log2MinCbSize][y0 >> Log2MinCbSize]	
if(slice_type != I)	
skip_flag[x0 **][** y0 **]**	ae(v)
if(skip_flag[x0][y0])	
prediction_unit(x0, y0 , log2CbSize)	
else if(slice_type != I || log2CbSize = = Log2MinCbSize) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(PredMode == MODE_INTRA && enable_intra_pred_flag = = 1 && enable_inter_layer_flag ==1)	
inter_layer_flag	ae(1)
if((PredMode != MODE_INTRA || log2CbSize = = Log2MinCbSize) && inter_layer_flag == 0)	
part_mode	ae(v)
if(PredMode != MODE_INTRA && **enable_grilp_flag = = 1**)	
grilp_flag	ae(v)
x1 = x0 + ((1 << log2CbSize) >> 1)	
y1 = y0 + ((1 << log2CbSize) >> 1)	
x2 = x1 – ((1 << log2CbSize) >> 2)	
y2 = y1 – ((1 << log2CbSize) >> 2)	
x3 = x1 + ((1 << log2CbSize) >> 2)	
y3 = y1 + ((1 << log2CbSize) >> 2)	
if(PartMode = = PART_2Nx2N)	
prediction_unit(x0, y0 , log2CbSize)	
else if(PartMode = = PART_2NxN) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x0, y1 , log2CbSize)	
} else if(PartMode = = PART_Nx2N) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x1, y0 , log2CbSize)	
} else if(PartMode = = PART_2NxnU) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x0, y2 , log2CbSize)	
} else if(PartMode = = PART_2NxnD) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x0, y3 , log2CbSize)	
} else if(PartMode = = PART_nLx2N) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x2, y0 , log2CbSize)	
} else if(PartMode = = PART_nRx2N) {	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x3, y0 , log2CbSize)	
} else { /* PART_NxN */	
prediction_unit(x0, y0 , log2CbSize)	
prediction_unit(x1, y0 , log2CbSize)	
prediction_unit(x0, y1 , log2CbSize)	
prediction_unit(x1, y1 , log2CbSize)	
}	
if(!pcm_flag)	
transform_tree(x0, y0, x0, y0, log2CbSize, log2CbSize, log2CbSize, 0, 0)	
}	
}	

prediction_unit(x0, y0, log2CbSize) {	**Descriptor**
if(skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else if(PredMode = = MODE_INTRA) {	
if(PartMode = = PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIPCMCUSize && log2CbSize <= Log2MaxIPCMCUSize)	
pcm_flag	ae(v)
if(pcm_flag) {	
num_subsequent_pcm	tu(3)
NumPCMBlock = num_subsequent_pcm + 1	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	u(v)
pcm_sample(x0, y0, log2CbSize)	
} else {	
if (enable_intra_pred_flag = = 0 || inter_layer_flag = = 1) {	
inter_layer_mode	ae(1)
} else {	
prev_intra_luma_pred_flag[x0 **][** y0 **]**	ae(v)
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_idx[x0 **][** y0 **]**	ae(v)
else	
rem_intra_luma_pred_mode[x0 **][** y0 **]**	ae(v)
intra_chroma_pred_mode[x0][y0]	ae(v)
SignalledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0][y0] = = 3 : intra_chroma_pred_mode[x0][y0] = = 2)	
}	
}	
} else { /* MODE_INTER */	
merge_flag[x0 **][** y0 **]**	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else {	
if(slice_type = = B)	
inter_pred_flag[x0 **][** y0 **]**	ae(v)
if(inter_pred_flag[x0][y0] = = Pred_LC) {	
if(num_ref_idx_lc_active_minus1 > 0)	
ref_idx_lc[x0 **][** y0 **]**	ae(v)
mvd_coding(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])	
mvp_lc_flag[x0][y0]	ae(v)

} else { /* Pred_L0 or Pred_BI */	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(mvd_l0[x0][y0][0], mvd_l0[x0][y0][1])	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_flag[x0][y0] = = Pred_BI) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0 **][** y0 **]**	ae(v)
if(mvd_l1_zero_flag) {	
mvd_l1[x0][y0][0] = 0	
mvd_l1[x0][y0][1] = 0	
} else	
mvd_coding(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])	
mvp_l1_flag[x0][y0]	ae(v)
}	
}	
}	
}	

The following diagrams in Figure 37 schematically illustrate how the different modes (Grilp, IntraBL and BaseMode) are coded in the bitstream. The coding actually depends on the enabled modes for the slice, signaled by the three flags enable_intra_pred_flag, enable_inter_layer_flag and enable_grilp_flag.

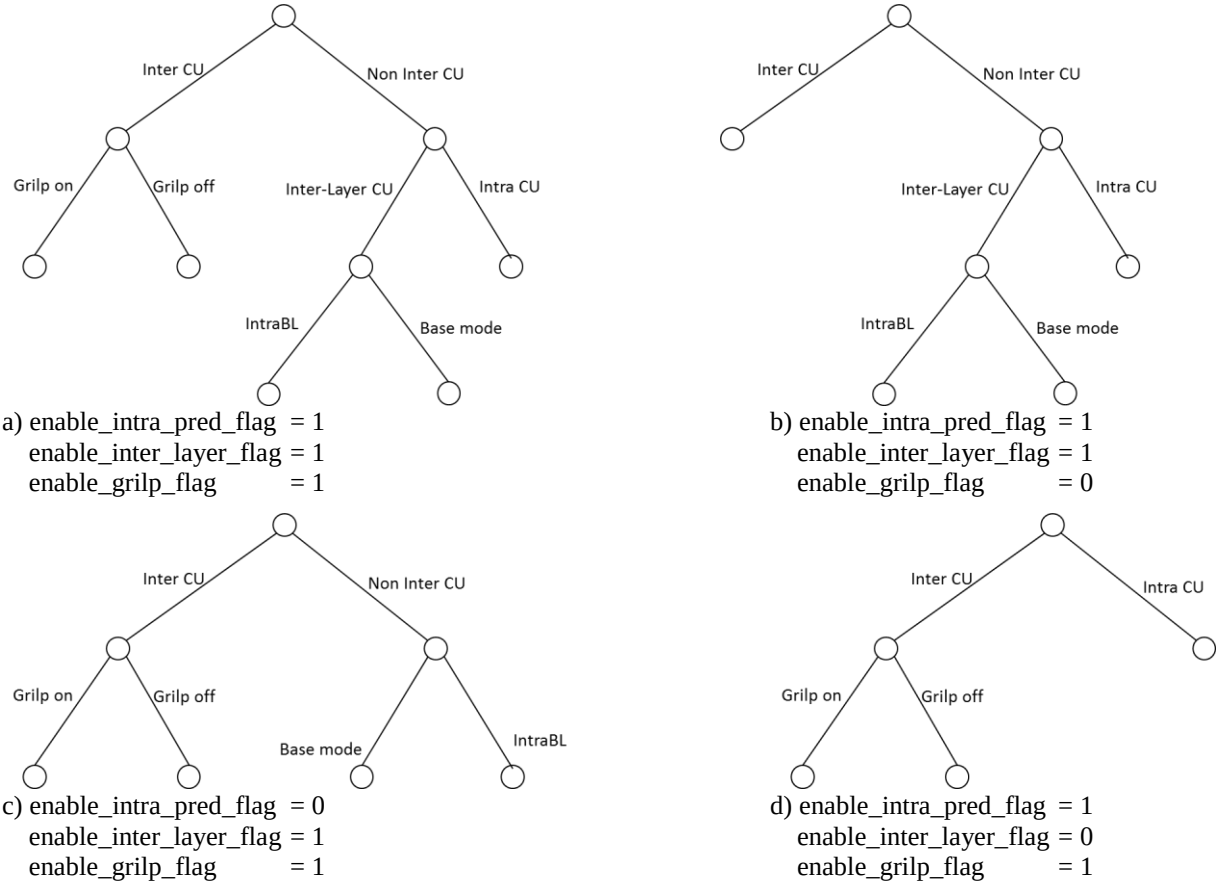

Figure 37: Semantic tree used to encode a Coding Unit mode

4 Compression performance discussion

4.1 Category 1 (HEVC base layer)

In this section, for each configuration of Category 1, two tables of performances are reported.
The first one results from PSNR computation with 10bit precision, provided by the HM6.1 encoder. The second table corresponds to PSNR values obtained with output decoded YUV sequences, represented with 8bit precision level. Exactly the same bit-streams are used to generate these two types of results. Indeed, the CSC codec has been designed so as to support 10bit internal processing, hence is able to handle 10bit video sequences.

4.1.1 Spatial scalability

4.1.1.1 Temporal Random Access configurations RA HEVC 2x

Class A	-37,9%	-27,6%	-27,8%
Class B	-34,2%	-27,4%	-26,0%
All	-36,0%	-27,5%	-26,9%
Traffic	-34,3%	-28,4%	-27,3%
PeopleOnStreet	-41,5%	-26,7%	-28,3%
Kimono	-43,2%	-40,0%	-37,9%
ParkScene	-31,6%	-26,5%	-23,6%
Cactus	-35,9%	-24,3%	-23,8%
BasketballDrive	-35,8%	-30,0%	-33,3%
BQTerrace	-24,4%	-16,3%	-11,2%
Delta -2	-22,3%	-16,0%	-15,2%
Delta 0	-28,2%	-21,4%	-19,7%
Delta 2	-36,5%	-30,5%	-30,1%
Delta 4	-52,5%	-46,8%	-45,4%

Class A	-37,1%	-23,7%	-22,6%
Class B	-33,0%	-22,7%	-19,9%
All	-35,0%	-23,2%	-21,3%
Traffic	-33,1%	-25,2%	-23,0%
PeopleOnStreet	-41,0%	-22,1%	-22,2%
Kimono	-41,7%	-35,9%	-33,1%
ParkScene	-30,7%	-24,0%	-20,5%
Cactus	-34,8%	-20,3%	-19,1%
BasketballDrive	-34,6%	-23,6%	-28,7%
BQTerrace	-23,2%	-9,7%	1,7%
Delta -2	-20,8%	-10,3%	-7,7%
Delta 0	-27,1%	-16,8%	-13,7%
Delta 2	-35,8%	-27,4%	-26,1%
Delta 4	-52,1%	-44,8%	-42,7%

Table 2: Codec performance enhancement layer vs. monocast, RA2x (left:10bit PSNR, right: 8bit PSNR)

Class A	7,3%	21,7%	21,4%
Class B	5,4%	13,4%	15,2%
All	6,3%	17,6%	18,3%
Traffic	9,2%	16,6%	17,7%
PeopleOnStreet	5,3%	26,9%	25,1%
Kimono	0,9%	4,2%	7,1%
ParkScene	9,0%	14,7%	17,7%
Cactus	5,5%	19,2%	20,2%
BasketballDrive	5,6%	13,1%	8,6%
BQTerrace	5,8%	15,9%	22,6%
Delta -2	-0,1%	8,1%	9,0%
Delta 0	4,2%	13,9%	15,9%
Delta 2	8,6%	18,2%	18,7%
Delta 4	10,8%	23,5%	26,1%

Class A	8,4%	26,7%	28,1%
Class B	6,9%	19,5%	22,9%
All	7,6%	23,1%	25,5%
Traffic	10,8%	20,5%	23,1%
PeopleOnStreet	6,0%	33,0%	33,0%
Kimono	3,0%	10,2%	14,1%
ParkScene	10,1%	17,7%	21,6%
Cactus	6,9%	24,4%	26,3%
BasketballDrive	7,2%	21,4%	14,6%
BQTerrace	7,2%	23,8%	38,1%
Delta -2	1,9%	15,2%	18,2%
Delta 0	5,8%	20,3%	24,2%
Delta 2	9,8%	23,4%	25,3%
Delta 4	11,7%	28,2%	32,1%

Table 3: Codec performance scalable (base+enh) vs. monocast, RA2x (left:10bit PSNR, right: 8bit PSNR)

Class A	-27,0%	-17,9%	-18,1%
Class B	-24,6%	-18,8%	-17,5%
All	-25,8%	-18,3%	-17,8%
Traffic	-24,2%	-19,1%	-18,2%
PeopleOnStreet	-29,8%	-16,6%	-17,9%
Kimono	-31,6%	-29,1%	-27,2%
ParkScene	-22,2%	-18,1%	-15,8%
Cactus	-25,7%	-16,0%	-15,5%
BasketballDrive	-25,7%	-20,7%	-23,5%
BQTerrace	-17,7%	-10,3%	-5,3%
Delta -2	-20,1%	-13,8%	-13,0%
Delta 0	-23,1%	-16,1%	-14,5%
Delta 2	-27,3%	-20,7%	-20,3%
Delta 4	-33,5%	-25,9%	-24,2%

Class A	-26,2%	-14,6%	-13,7%
Class B	-23,5%	-14,6%	-12,1%
All	-24,9%	-14,6%	-12,9%
Traffic	-23,1%	-16,4%	-14,5%
PeopleOnStreet	-29,3%	-12,7%	-12,8%
Kimono	-30,3%	-25,3%	-22,8%
ParkScene	-21,4%	-16,0%	-13,1%
Cactus	-24,7%	-12,5%	-11,4%
BasketballDrive	-24,6%	-15,1%	-19,5%
BQTerrace	-16,6%	-4,3%	6,3%
Delta -2	-18,5%	-8,2%	-5,7%
Delta 0	-22,0%	-11,5%	-8,5%
Delta 2	-26,5%	-17,3%	-16,0%
Delta 4	-33,0%	-23,1%	-20,7%

Table 4: Codec performance scalable (base+enh) vs. simulcast, RA2x (left:10bit PSNR, right: 8bit PSNR)

4.1.1.2 Temporal Random Access configurations RA HEVC 1.5x

Class A			
Class B	-53,1%	-43,3%	-40,4%
All	-53,1%	-43,3%	-40,4%
Traffic			
PeopleOnStreet			
Kimono	-60,2%	-56,4%	-53,8%
ParkScene	-52,4%	-46,9%	-42,3%
Cactus	-54,7%	-44,6%	-42,3%
BasketballDrive	-54,9%	-47,7%	-49,2%
BQTerrace	-43,3%	-21,0%	-14,4%
Delta -2	-30,6%	-19,9%	-14,5%
Delta 0	-41,2%	-31,8%	-27,2%
Delta 2	-61,3%	-56,8%	-57,5%
Delta 4	-78,1%	-76,5%	-77,1%

Class A			
Class B	-52,1%	-39,4%	-34,8%
All	-52,1%	-39,4%	-34,8%
Traffic			
PeopleOnStreet			
Kimono	-58,9%	-52,9%	-49,7%
ParkScene	-51,7%	-45,0%	-40,0%
Cactus	-53,8%	-41,3%	-38,3%
BasketballDrive	-53,8%	-43,3%	-45,9%
BQTerrace	-42,3%	-14,6%	-0,1%
Delta -2	-29,0%	-14,2%	-6,4%
Delta 0	-40,2%	-27,6%	-21,5%
Delta 2	-60,9%	-54,8%	-55,2%
Delta 4	-77,9%	-75,5%	-76,0%

Table 5: Codec performance enhancement layer vs. monocast, RA1.5x (left:10bit PSNR, right: 8bit PSNR)

Class A			
Class B	4,1%	17,0%	21,1%
All	4,1%	17,0%	21,1%
Traffic			
PeopleOnStreet			
Kimono	-0,2%	3,0%	7,1%
ParkScene	8,1%	15,1%	20,8%
Cactus	4,1%	17,9%	23,7%
BasketballDrive	3,1%	13,4%	11,3%
BQTerrace	5,5%	35,6%	42,4%
Delta -2	-0,4%	14,0%	20,9%
Delta 0	3,9%	19,2%	26,0%
Delta 2	6,7%	16,9%	15,8%
Delta 4	6,8%	13,6%	11,3%

Class A			
Class B	5,7%	23,4%	29,3%
All	5,7%	23,4%	29,3%
Traffic			
PeopleOnStreet			
Kimono	1,9%	9,1%	14,0%
ParkScene	9,2%	18,3%	24,7%
Cactus	5,6%	23,3%	30,2%
BasketballDrive	4,8%	21,8%	17,5%
BQTerrace	7,0%	44,5%	60,3%
Delta -2	1,7%	21,9%	31,6%
Delta 0	5,6%	26,2%	35,2%
Delta 2	7,9%	22,1%	21,7%
Delta 4	7,8%	18,3%	16,4%

Table 6: Codec performance scalable (base+enh) vs. monocast, RA1.5x (left:10bit PSNR, right: 8bit PSNR)

Class A			
Class B	-33,7%	-25,6%	-23,1%
All	-33,7%	-25,6%	-23,1%
Traffic			
PeopleOnStreet			
Kimono	-39,3%	-36,8%	-34,5%
ParkScene	-32,5%	-28,1%	-24,5%
Cactus	-34,7%	-26,2%	-23,5%
BasketballDrive	-35,0%	-28,8%	-30,0%
BQTerrace	-27,0%	-7,9%	-3,0%
Delta -2	-25,6%	-15,0%	-9,7%
Delta 0	-30,5%	-20,2%	-15,6%
Delta 2	-37,5%	-31,1%	-31,8%
Delta 4	-45,6%	-42,0%	-43,2%

Class A			
Class B	-32,8%	-21,9%	-18,3%
All	-32,8%	-21,9%	-18,3%
Traffic			
PeopleOnStreet			
Kimono	-38,1%	-33,4%	-30,6%
ParkScene	-31,8%	-26,3%	-22,3%
Cactus	-33,8%	-23,2%	-20,0%
BasketballDrive	-34,1%	-24,4%	-26,7%
BQTerrace	-26,0%	-2,3%	8,3%
Delta -2	-24,1%	-9,3%	-1,9%
Delta 0	-29,4%	-15,7%	-9,6%
Delta 2	-36,7%	-28,2%	-28,5%
Delta 4	-45,1%	-39,6%	-40,6%

Table 7: Codec performance scalable (base+enh) vs. simulcast, RA1.5x (left:10bit PSNR, right: 8bit PSNR)

4.1.2 Intra-only spatial scalability

4.1.2.1 All-Intra configuration AI HEVC 2x

Class A	-40,1%	-41,9%	-42,7%
Class B	-33,3%	-28,6%	-29,6%
All	-36,7%	-35,2%	-36,1%
Traffic	-38,4%	-39,4%	-39,8%
PeopleOnStreet	-41,9%	-44,4%	-45,6%
Kimono	-45,1%	-45,2%	-44,0%
ParkScene	-35,7%	-36,3%	-35,6%
Cactus	-31,8%	-25,6%	-31,1%
BasketballDrive	-27,8%	-21,5%	-26,8%
BQTerrace	-26,0%	-14,5%	-10,2%
Delta -2	-21,5%	-19,4%	-19,8%
Delta 0	-29,0%	-27,0%	-26,7%
Delta 2	-38,6%	-36,5%	-37,7%
Delta 4	-52,4%	-50,9%	-52,3%

Class A	-39,4%	-40,2%	-40,4%
Class B	-32,3%	-26,4%	-26,8%
All	-35,9%	-33,3%	-33,6%
Traffic	-37,5%	-38,1%	-37,9%
PeopleOnStreet	-41,3%	-42,3%	-43,0%
Kimono	-43,5%	-42,7%	-41,1%
ParkScene	-34,9%	-35,1%	-33,9%
Cactus	-31,1%	-23,9%	-28,8%
BasketballDrive	-26,6%	-17,7%	-23,6%
BQTerrace	-25,4%	-12,5%	-6,8%
Delta -2	-20,2%	-16,7%	-16,4%
Delta 0	-28,1%	-24,9%	-24,0%
Delta 2	-38,0%	-35,1%	-36,0%
Delta 4	-52,1%	-49,8%	-51,1%

Table 8: Codec performance enhancement layer vs. monocast, AI2x (left:10bit PSNR, right: 8bit PSNR)

Class A	7,6%	4,2%	3,1%
Class B	9,6%	15,4%	14,1%
All	8,6%	9,8%	8,6%
Traffic	9,0%	6,6%	5,9%
PeopleOnStreet	6,3%	1,9%	0,2%
Kimono	1,3%	-0,5%	1,0%
ParkScene	5,7%	4,0%	4,5%
Cactus	12,3%	19,8%	13,3%
BasketballDrive	17,0%	26,4%	19,2%
BQTerrace	12,0%	27,4%	32,6%
Delta -2	7,0%	10,1%	9,7%
Delta 0	8,8%	11,6%	12,1%
Delta 2	10,4%	13,6%	11,7%
Delta 4	10,0%	13,0%	9,8%

Class A	8,7%	6,6%	6,2%
Class B	11,0%	18,5%	17,9%
All	9,8%	12,6%	12,1%
Traffic	10,2%	8,4%	8,4%
PeopleOnStreet	7,2%	4,9%	4,1%
Kimono	3,7%	3,2%	5,3%
ParkScene	6,6%	5,5%	6,7%
Cactus	13,4%	22,1%	16,6%
BasketballDrive	18,7%	31,6%	23,7%
BQTerrace	12,8%	30,1%	37,1%
Delta -2	8,7%	13,7%	14,3%
Delta 0	10,1%	14,9%	16,1%
Delta 2	11,5%	16,1%	14,7%
Delta 4	10,9%	15,3%	12,5%

Table 9: Codec performance scalable (base+enh) vs. monocast, AI2x (left:10bit PSNR, right: 8bit PSNR)

Class A	-27,8%	-29,7%	-30,4%
Class B	-22,9%	-19,0%	-19,8%
All	-25,3%	-24,3%	-25,1%
Traffic	-26,5%	-27,6%	-28,0%
PeopleOnStreet	-29,1%	-31,7%	-32,8%
Kimono	-32,2%	-32,8%	-31,7%
ParkScene	-25,1%	-25,9%	-25,4%
Cactus	-21,4%	-16,4%	-20,9%
BasketballDrive	-18,0%	-12,6%	-17,0%
BQTerrace	-17,6%	-7,3%	-3,8%
Delta -2	-17,5%	-15,5%	-15,9%
Delta 0	-21,7%	-19,7%	-19,4%
Delta 2	-26,7%	-24,5%	-25,8%
Delta 4	-33,1%	-31,3%	-33,1%

Class A	-27,1%	-28,1%	-28,4%
Class B	-22,0%	-17,0%	-17,4%
All	-24,6%	-22,6%	-22,9%
Traffic	-25,7%	-26,5%	-26,4%
PeopleOnStreet	-28,6%	-29,8%	-30,4%
Kimono	-30,7%	-30,5%	-29,0%
ParkScene	-24,5%	-24,9%	-23,9%
Cactus	-20,8%	-14,9%	-18,8%
BasketballDrive	-16,9%	-9,3%	-14,3%
BQTerrace	-17,0%	-5,5%	-0,8%
Delta -2	-16,3%	-12,8%	-12,5%
Delta 0	-20,7%	-17,5%	-16,6%
Delta 2	-26,0%	-22,9%	-23,9%
Delta 4	-32,6%	-29,9%	-31,5%

Table 10: Codec performance scalable (base+enh) vs. simulcast, AI2x (left:10bit PSNR, right: 8bit PSNR)

4.1.2.2 All-Intra configuration AI HEVC 1.5x

Class A			
Class B	-56,8%	-52,6%	-53,1%
All	-56,8%	-52,6%	-53,1%
Traffic			
PeopleOnStreet			
Kimono	-64,4%	-63,0%	-61,8%
ParkScene	-58,6%	-57,1%	-59,8%
Cactus	-56,3%	-50,7%	-54,8%
BasketballDrive	-53,3%	-48,2%	-51,0%
BQTerrace	-51,3%	-43,9%	-38,2%
Delta -2	-34,0%	-32,2%	-31,0%
Delta 0	-46,7%	-44,8%	-44,9%
Delta 2	-65,7%	-64,5%	-65,3%
Delta 4	-81,7%	-81,9%	-82,4%

Class A			
Class B	-56,0%	-50,9%	-51,1%
All	-56,0%	-50,9%	-51,1%
Traffic			
PeopleOnStreet			
Kimono	-63,0%	-60,9%	-59,5%
ParkScene	-58,1%	-56,1%	-59,0%
Cactus	-55,7%	-49,6%	-53,2%
BasketballDrive	-52,4%	-45,3%	-48,8%
BQTerrace	-50,8%	-42,5%	-34,9%
Delta -2	-32,9%	-29,9%	-28,1%
Delta 0	-46,0%	-43,2%	-43,0%
Delta 2	-65,3%	-63,7%	-64,5%
Delta 4	-81,6%	-81,6%	-82,0%

Table 11: Codec performance enhancement layer vs. monocast, AI1.5x (left:10bit PSNR, right: 8bit PSNR)

Class A			
Class B	5,7%	10,2%	10,4%
All	5,7%	10,2%	10,4%
Traffic			
PeopleOnStreet			
Kimono	-0,4%	-1,3%	0,2%
ParkScene	3,8%	3,1%	2,5%
Cactus	7,2%	11,8%	8,7%
BasketballDrive	10,2%	19,6%	15,3%
BQTerrace	7,9%	17,7%	25,5%
Delta -2	5,8%	9,9%	12,1%
Delta 0	6,6%	10,9%	11,2%
Delta 2	6,0%	9,3%	6,6%
Delta 4	4,5%	2,6%	0,0%

Class A			
Class B	7,1%	13,1%	14,1%
All	7,1%	13,1%	14,1%
Traffic			
PeopleOnStreet			
Kimono	2,0%	2,3%	4,4%
ParkScene	4,7%	4,8%	4,5%
Cactus	8,3%	14,0%	11,8%
BasketballDrive	11,8%	24,2%	19,2%
BQTerrace	8,6%	20,1%	30,6%
Delta -2	7,6%	13,5%	16,6%
Delta 0	8,1%	14,0%	14,9%
Delta 2	7,0%	11,5%	9,1%
Delta 4	5,3%	4,7%	2,2%

Table 12: Codec performance scalable (base+enh) vs. monocast, AI1.5x (left:10bit PSNR, right: 8bit PSNR)

Class A			
Class B	-34,3%	-31,6%	-31,4%
All	-34,3%	-31,6%	-31,4%
Traffic			
PeopleOnStreet			
Kimono	-40,3%	-40,3%	-39,3%
ParkScene	-35,8%	-35,6%	-35,9%
Cactus	-33,7%	-30,4%	-32,9%
BasketballDrive	-31,2%	-27,0%	-29,1%
BQTerrace	-30,6%	-24,9%	-19,8%
Delta -2	-24,9%	-22,7%	-21,5%
Delta 0	-31,0%	-28,6%	-28,6%
Delta 2	-38,5%	-36,6%	-38,0%
Delta 4	-45,6%	-46,4%	-47,7%

Class A			
Class B	-33,6%	-30,1%	-29,4%
All	-33,6%	-30,1%	-29,4%
Traffic			
PeopleOnStreet			
Kimono	-39,0%	-38,2%	-37,0%
ParkScene	-35,3%	-34,7%	-34,7%
Cactus	-33,2%	-29,3%	-31,3%
BasketballDrive	-30,3%	-24,6%	-27,1%
BQTerrace	-30,1%	-23,6%	-17,1%
Delta -2	-23,7%	-20,3%	-18,4%
Delta 0	-30,1%	-26,7%	-26,3%
Delta 2	-37,9%	-35,3%	-36,6%
Delta 4	-45,2%	-45,4%	-46,6%

Table 13: Codec performance scalable (base+enh) vs. simulcast, AI1.5x (left:10bit PSNR, right: 8bit PSNR)

4.1.3 SNR scalability

4.1.3.1 Temporal Random Access configurations RA HEVC SNR

Class A	-45,3%	-38,4%	-38,6%
Class B	-42,2%	-36,9%	-35,7%
All	-43,8%	-37,6%	-37,2%
Traffic	-42,7%	-37,3%	-35,9%
PeopleOnStreet	-47,9%	-39,5%	-41,3%
Kimono	-46,5%	-46,1%	-44,1%
ParkScene	-40,6%	-38,9%	-38,0%
Cactus	-43,3%	-35,2%	-30,8%
BasketballDrive	-43,9%	-37,4%	-38,5%
BQTerrace	-36,9%	-26,6%	-27,2%
Delta -2	-24,8%	-19,0%	-17,8%
Delta 0	-32,0%	-27,3%	-26,4%
Delta 2	-43,8%	-38,4%	-38,5%
Delta 4	-67,4%	-66,4%	-66,7%

Class A	-44,2%	-34,2%	-32,9%
Class B	-40,9%	-31,5%	-28,6%
All	-42,5%	-32,9%	-30,8%
Traffic	-41,2%	-33,9%	-30,8%
PeopleOnStreet	-47,3%	-34,6%	-35,0%
Kimono	-44,7%	-41,5%	-38,7%
ParkScene	-39,5%	-36,0%	-34,5%
Cactus	-41,9%	-31,0%	-24,7%
BasketballDrive	-42,5%	-29,9%	-32,5%
BQTerrace	-35,6%	-19,0%	-12,5%
Delta -2	-22,8%	-12,1%	-8,2%
Delta 0	-30,6%	-22,2%	-19,7%
Delta 2	-42,9%	-34,9%	-33,9%
Delta 4	-67,0%	-65,0%	-64,9%

Table 14: Codec performance enhancement layer vs. monocast, RA SNR (left:10bit PSNR, right: 8bit PSNR)

Class A	2,0%	11,5%	11,1%
Class B	0,9%	7,4%	8,8%
All	1,4%	9,4%	9,9%
Traffic	4,2%	11,3%	13,0%
PeopleOnStreet	-0,2%	11,7%	9,1%
Kimono	0,7%	0,7%	3,1%
ParkScene	6,5%	8,4%	9,1%
Cactus	0,4%	9,6%	16,7%
BasketballDrive	0,2%	8,0%	6,3%
BQTerrace	-3,4%	10,3%	8,6%
Delta -2	-2,9%	4,3%	6,3%
Delta 0	0,4%	7,5%	9,0%
Delta 2	3,1%	12,5%	12,4%
Delta 4	2,7%	5,4%	4,7%

Class A	3,4%	17,1%	18,8%
Class B	2,8%	14,5%	18,1%
All	3,1%	15,8%	18,5%
Traffic	6,2%	15,7%	19,5%
PeopleOnStreet	0,7%	18,4%	18,0%
Kimono	3,3%	7,2%	10,6%
ParkScene	8,0%	12,1%	13,5%
Cactus	2,3%	15,0%	24,9%
BasketballDrive	2,1%	18,3%	14,4%
BQTerrace	-1,8%	19,9%	27,4%
Delta -2	-0,5%	12,9%	18,0%
Delta 0	2,4%	14,6%	18,3%
Delta 2	4,6%	18,9%	20,6%
Delta 4	3,8%	9,9%	10,3%

Table 15: Codec performance scalable (base+enh) vs. monocast, RA SNR (left:10bit PSNR, right: 8bit PSNR)

Class A	-31,2%	-25,2%	-25,4%
Class B	-29,4%	-24,8%	-23,8%
All	-30,3%	-25,0%	-24,6%
Traffic	-29,1%	-24,4%	-23,3%
PeopleOnStreet	-33,3%	-25,9%	-27,5%
Kimono	-32,4%	-32,0%	-30,4%
ParkScene	-27,4%	-26,1%	-25,4%
Cactus	-30,1%	-23,5%	-19,2%
BasketballDrive	-30,6%	-25,2%	-26,2%
BQTerrace	-26,5%	-17,0%	-18,0%
Delta -2	-22,2%	-16,6%	-15,3%
Delta 0	-26,5%	-21,6%	-20,7%
Delta 2	-32,4%	-26,4%	-26,4%
Delta 4	-40,9%	-39,3%	-39,7%

Class A	-30,3%	-21,6%	-20,5%
Class B	-28,1%	-20,0%	-17,5%
All	-29,2%	-20,8%	-19,0%
Traffic	-27,8%	-21,5%	-19,0%
PeopleOnStreet	-32,7%	-21,6%	-22,0%
Kimono	-30,7%	-27,8%	-25,5%
ParkScene	-26,4%	-23,6%	-22,4%
Cactus	-28,8%	-19,8%	-14,0%
BasketballDrive	-29,3%	-18,5%	-20,9%
BQTerrace	-25,3%	-10,2%	-4,8%
Delta -2	-20,3%	-9,8%	-6,0%
Delta 0	-25,1%	-16,6%	-14,0%
Delta 2	-31,4%	-22,3%	-21,2%
Delta 4	-40,3%	-36,8%	-36,6%

Table 16: Codec performance scalable (base+enh) vs. simulcast, RA SNR (left:10bit PSNR, right: 8bit PSNR)

4.1.3.2 All-Intra configuration AI HEVC SNR (for information only)

Class A	-48,3%	-52,4%	-53,6%
Class B	-46,9%	-51,5%	-51,6%
All	-47,6%	-52,0%	-52,6%
Traffic	-46,7%	-49,7%	-49,7%
PeopleOnStreet	-49,8%	-55,2%	-57,5%
Kimono	-48,9%	-51,0%	-49,2%
ParkScene	-44,7%	-48,0%	-48,2%
Cactus	-43,1%	-44,9%	-46,0%
BasketballDrive	-48,0%	-53,2%	-50,8%
BQTerrace	-49,9%	-60,3%	-63,9%
Delta -2	-26,6%	-31,0%	-31,1%
Delta 0	-35,8%	-40,3%	-40,1%
Delta 2	-48,1%	-51,7%	-52,2%
Delta 4	-70,3%	-72,0%	-72,2%

Class A	-47,3%	-50,5%	-51,1%
Class B	-45,8%	-49,2%	-48,6%
All	-46,6%	-49,8%	-49,9%
Traffic	-45,6%	-48,2%	-47,6%
PeopleOnStreet	-49,0%	-52,8%	-54,7%
Kimono	-47,0%	-48,0%	-45,6%
ParkScene	-43,9%	-46,6%	-46,5%
Cactus	-42,3%	-43,2%	-43,3%
BasketballDrive	-46,8%	-49,5%	-47,3%
BQTerrace	-49,3%	-58,6%	-60,4%
Delta -2	-25,1%	-28,0%	-27,1%
Delta 0	-34,8%	-38,0%	-37,3%
Delta 2	-47,4%	-50,3%	-50,4%
Delta 4	-70,1%	-71,4%	-71,4%

Table 17: Codec performance enhancement layer vs. monocast, AI SNR (left:10bit PSNR, right: 8bit PSNR)

Class A	5,0%	-0,9%	-2,8%
Class B	5,9%	-1,2%	-1,4%
All	5,4%	-1,1%	-2,1%
Traffic	6,8%	2,8%	2,5%
PeopleOnStreet	3,1%	-4,6%	-8,0%
Kimono	1,9%	-1,0%	1,4%
ParkScene	5,3%	1,0%	0,6%
Cactus	6,7%	3,6%	3,5%
BasketballDrive	8,9%	1,8%	5,8%
BQTerrace	6,8%	-11,6%	-18,2%
Delta -2	6,1%	0,2%	0,3%
Delta 0	6,4%	-0,8%	-0,3%
Delta 2	6,1%	-1,5%	-2,4%
Delta 4	2,9%	-3,2%	-3,8%

Class A	6,4%	2,1%	1,1%
Class B	7,6%	2,3%	3,2%
All	7,0%	2,2%	2,1%
Traffic	8,5%	5,1%	5,6%
PeopleOnStreet	4,3%	-0,8%	-3,5%
Kimono	4,9%	3,5%	6,7%
ParkScene	6,5%	3,1%	3,3%
Cactus	7,9%	6,0%	7,5%
BasketballDrive	10,7%	7,8%	11,5%
BQTerrace	7,8%	-9,0%	-13,0%
Delta -2	8,3%	4,6%	6,0%
Delta 0	8,2%	2,8%	4,2%
Delta 2	7,4%	1,5%	1,4%
Delta 4	3,9%	-1,0%	-1,2%

Table 18: Codec performance scalable (base+enh) vs. monocast, AI SNR (left:10bit PSNR, right: 8bit PSNR)

Class A	-32,8%	-36,5%	-37,6%
Class B	-31,8%	-36,1%	-36,2%
All	-32,3%	-36,3%	-36,9%
Traffic	-31,5%	-34,1%	-34,2%
PeopleOnStreet	-34,1%	-39,0%	-41,1%
Kimono	-33,7%	-35,5%	-33,9%
ParkScene	-30,6%	-33,4%	-33,6%
Cactus	-29,0%	-30,8%	-31,4%
BasketballDrive	-32,1%	-36,7%	-34,4%
BQTerrace	-33,7%	-44,1%	-47,8%
Delta -2	-23,4%	-27,8%	-27,9%
Delta 0	-29,4%	-34,1%	-33,9%
Delta 2	-36,1%	-40,5%	-41,1%
Delta 4	-44,3%	-47,5%	-47,9%

Class A	-32,0%	-34,7%	-35,3%
Class B	-30,8%	-34,0%	-33,5%
All	-31,4%	-34,3%	-34,4%
Traffic	-30,5%	-32,7%	-32,3%
PeopleOnStreet	-33,4%	-36,7%	-38,3%
Kimono	-31,9%	-32,7%	-30,6%
ParkScene	-29,8%	-32,1%	-32,0%
Cactus	-28,2%	-29,2%	-28,9%
BasketballDrive	-31,0%	-33,2%	-31,2%
BQTerrace	-33,1%	-42,5%	-44,6%
Delta -2	-21,8%	-24,7%	-23,8%
Delta 0	-28,3%	-31,8%	-31,0%
Delta 2	-35,3%	-38,8%	-38,9%
Delta 4	-43,8%	-46,3%	-46,4%

Table 19: Codec performance scalable (base+enh) vs. simulcast, AI SNR (left:10bit PSNR, right: 8bit PSNR)

4.1.4 Overall

enhancement vs. monocast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-40,1%	-41,9%	-42,7%			
Class B	-33,3%	-28,6%	-29,6%	-56,8%	-52,6%	-53,1%
Overall	-36,7%	-35,2%	-36,1%	-56,8%	-52,6%	-53,1%
	-36,7%	-35,1%	-36,3%	-56,9%	-55,4%	-54,2%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-37,9%	-27,6%	-27,8%			
Class B	-34,2%	-27,4%	-26,0%	-53,1%	-43,3%	-40,4%
Overall	-36,0%	-27,5%	-26,9%	-53,1%	-43,3%	-40,4%
	-36,1%	-27,5%	-26,9%	-53,2%	-42,7%	-38,9%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	-45,3%	-38,4%	-38,6%	-48,3%	-52,4%	-53,6%
Class B	-42,2%	-36,9%	-35,7%	-46,9%	-51,5%	-51,6%
Overall	-43,8%	-37,6%	-37,2%	-47,6%	-52,0%	-52,6%
	-43,7%	-37,6%	-37,2%	-47,6%	-52,1%	-52,7%

	Y	U	V
Overall AI	-46,7%	-43,9%	-44,6%
Overall RA	-44,6%	-35,4%	-33,6%
Overall RA SNR	-43,8%	-37,6%	-37,2%

enhancement vs. monocast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-39,4%	-40,2%	-40,4%			
Class B	-32,3%	-26,4%	-26,8%	-56,0%	-50,9%	-51,1%
Overall	-35,9%	-33,3%	-33,6%	-56,0%	-50,9%	-51,1%
	-35,9%	-33,2%	-34,0%	-56,2%	-53,9%	-52,3%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-37,1%	-23,7%	-22,6%			
Class B	-33,0%	-22,7%	-19,9%	-52,1%	-39,4%	-34,8%
Overall	-35,0%	-23,2%	-21,3%	-52,1%	-39,4%	-34,8%
	-35,0%	-23,2%	-21,4%	-52,2%	-38,9%	-34,3%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	-44,2%	-34,2%	-32,9%	-47,3%	-50,5%	-51,1%
Class B	-40,9%	-31,5%	-28,6%	-45,8%	-49,2%	-48,6%
Overall	-42,5%	-32,9%	-30,8%	-46,6%	-49,8%	-49,9%
	-42,5%	-32,9%	-30,8%	-46,5%	-49,9%	-50,0%

	Y	U	V
Overall AI	-45,9%	-42,1%	-42,4%
Overall RA	-43,6%	-31,3%	-28,0%
Overall RA SNR	-42,5%	-32,9%	-30,8%

Table 20: Summary of performance enhancement layer vs. monocast (left:10bit PSNR, right: 8bit PSNR)

scalable vs. monocast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	7,6%	4,2%	7,6%			
Class B	9,6%	15,4%	14,1%	5,7%	10,2%	10,4%
Overall	8,6%	9,8%	8,6%	5,7%	10,2%	10,4%
	8,6%	9,9%	8,4%	5,7%	5,7%	5,8%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	7,3%	21,7%	21,4%			
Class B	5,4%	13,4%	15,2%	4,1%	17,0%	21,1%
Overall	6,3%	17,6%	18,3%	4,1%	17,0%	21,1%
	6,3%	17,6%	18,3%	4,1%	17,3%	21,5%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	2,0%	11,5%	11,1%	5,0%	-0,9%	-2,8%
Class B	0,9%	7,4%	8,8%	5,9%	-1,2%	-1,4%
Overall	1,4%	9,4%	9,9%	5,4%	-1,1%	-2,1%
	1,4%	9,4%	9,8%	5,4%	-1,4%	-2,5%

	Y	U	V
Overall AI	7,2%	10,0%	9,5%
Overall RA	5,2%	17,3%	19,7%
Overall RA SNR	1,4%	9,4%	9,9%

scalable vs. monocast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	8,7%	6,6%	7,6%			
Class B	11,0%	18,5%	17,9%	7,1%	13,1%	14,1%
Overall	9,8%	12,6%	12,1%	7,1%	13,1%	14,1%
	9,8%	12,6%	11,7%	7,0%	8,5%	9,2%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	8,4%	26,7%	28,1%			
Class B	6,9%	19,5%	22,9%	5,7%	23,4%	29,3%
Overall	7,6%	23,1%	25,5%	5,7%	23,4%	29,3%
	7,6%	23,1%	25,5%	5,6%	23,6%	29,3%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	3,4%	17,1%	18,8%	6,4%	2,1%	1,1%
Class B	2,8%	14,5%	18,1%	7,6%	2,3%	3,2%
Overall	3,1%	15,8%	18,5%	7,0%	2,2%	2,1%
	3,1%	15,8%	18,4%	6,9%	1,9%	1,8%

	Y	U	V
Overall AI	8,5%	12,8%	13,1%
Overall RA	6,7%	23,3%	27,4%
Overall RA SNR	3,1%	15,8%	18,5%

Table 21: Summary of performance scalable (base+enh) vs. monocast (left:10bit PSNR, right: 8bit PSNR)

scalable vs. simulcast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-27,8%	-29,7%	-30,4%			
Class B	-22,9%	-19,0%	-19,8%	-34,3%	-31,6%	-31,4%
Overall	-25,3%	-24,3%	-25,1%	-34,3%	-31,6%	-31,4%
	-25,4%	-24,3%	-25,2%	-34,3%	-34,6%	-34,5%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-27,0%	-17,9%	-18,1%			
Class B	-24,6%	-18,8%	-17,5%	-33,7%	-25,6%	-23,1%
Overall	-25,8%	-18,3%	-17,8%	-33,7%	-25,6%	-23,1%
	-25,8%	-18,3%	-17,8%	-33,8%	-25,4%	-22,8%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	-31,2%	-25,2%	-25,4%	-32,8%	-36,5%	-37,6%
Class B	-29,4%	-24,8%	-23,8%	-31,8%	-36,1%	-36,2%
Overall	-30,3%	-25,0%	-24,6%	-32,3%	-36,3%	-36,9%
	-30,3%	-25,0%	-24,6%	-32,3%	-36,5%	-37,2%

	Y	U	V
Overall AI	-29,8%	-28,0%	-28,2%
Overall RA	-29,7%	-22,0%	-20,4%
Overall RA SNR	-30,3%	-25,0%	-24,6%

scalable vs. simulcast

	All Intra HEVC 2x			All Intra HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-27,1%	-28,1%	-28,4%			
Class B	-22,0%	-17,0%	-17,4%	-33,6%	-30,1%	-29,4%
Overall	-24,6%	-22,6%	-22,9%	-33,6%	-30,1%	-29,4%
	-24,6%	-22,5%	-23,1%	-33,6%	-33,1%	-32,6%

	Random Access HEVC 2x			Random Access HEVC 1.5x		
	Y	U	V	Y	U	V
Class A+	-26,2%	-14,6%	-13,7%			
Class B	-23,5%	-14,6%	-12,1%	-32,8%	-21,9%	-18,3%
Overall	-24,9%	-14,6%	-12,9%	-32,8%	-21,9%	-18,3%
	-24,9%	-14,6%	-12,9%	-32,8%	-21,8%	-18,3%

	Random Access HEVC SNR			All Intra HEVC SNR (informative)		
	Y	U	V	Y	U	V
Class A+	-30,3%	-21,6%	-20,5%	-32,0%	-34,7%	-35,3%
Class B	-28,1%	-20,0%	-17,5%	-30,8%	-34,0%	-33,5%
Overall	-29,2%	-20,8%	-19,0%	-31,4%	-34,3%	-34,4%
	-29,2%	-20,8%	-19,0%	-31,4%	-34,5%	-34,6%

	Y	U	V
Overall AI	-29,1%	-26,3%	-26,2%
Overall RA	-28,8%	-18,3%	-15,6%
Overall RA SNR	-29,2%	-20,8%	-19,0%

Table 22: Summary of performance scalable (base+enh) vs. simulcast (left:10bit PSNR, right: 8bit PSNR)

4.2 Category 2 (AVC base layer)

4.2.1 Spatial scalability

4.2.2 Overall

5 Complexity analysis

5.1 Encoding time and measurement methodology

The encoding times have been measured for each HEVC base layer configuration (Category 1 of section 4.1). Encoding times are provided for each coded stream listed in the attached performances Excel file.

For each given configuration, the coding times for scalable bit-streams and for anchor bit-streams have been measure strictly on the same platform.

5.2 Decoding time and measurement methodology and comparison vs. anchor bitstreams decoded by HM

(Comparison to HM6.1 should be with YUV output *enabled* and reference input *disabled*.)

Decoding times have been measured for each Category 1 configuration on the same platform, respectively for anchor and scalable bit-streams.

The encoding and decoding time, at bitrate targets corresponds to anchors with QP=22, are listed in the following tables.

			Scalable Enhan Tested					Anchors		
AI HEVC 2x	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchors	Enc T [h]	Enc T/Anchors	Enc T [s]	Dec T [s]	Enc T [h]
Traffic QPB 22	22	22	7963,56	234,91	131,03%	2,21	82,15%	9693,50	179,28	2,69
PeopleOnStreet QPB 22	22	22	9056,99	250,82	138,66%	2,52	86,04%	10526,45	180,89	2,92
Kimono QPB 22	22	22	2934,20	84,24	140,80%	0,82	79,96%	3669,80	59,83	1,02
ParkScene QPB 22	22	22	3627,55	109,20	132,40%	1,01	83,17%	4361,64	82,48	1,21
Cactus QPB 22	22	22	7414,56	227,00	139,79%	2,06	77,21%	9603,70	162,38	2,67
BasketballDrive QPB 22	22	22	6315,90	197,32	138,80%	1,75	71,32%	8856,15	142,17	2,46
BQTerrace QPB 22	22	22	8226,74	283,82	129,25%	2,29	67,93%	12110,62	219,59	3,36
Time geomean			6047,32	183,07	135,75%	1,68	78,01%	7752,02	134,86	2,15
Time sum (hours)			12,65	0,39	135,13%	12,65	77,42%	16,34	0,29	16,34

Table 23: comparative encoding and decoding times with anchor values (AI HEVC 2x)

			Scalable Enhan Tested					Anchors		
AI HEVC 1.5x	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchors	Enc T [h]	Enc T/Anchors	Enc T [s]	Dec T [s]	Enc T [h]
Kimono	22	22	3515,22	92,40	154,45%	0,98	95,79%	3669,80	59,83	1,02
ParkScene	22	22	4421,38	119,57	144,97%	1,23	101,37%	4361,64	82,48	1,21
Cactus	22	22	9082,61	242,84	149,55%	2,52	94,57%	9603,70	162,38	2,67
BasketballDrive	22	22	7941,34	217,50	152,99%	2,21	89,67%	8856,15	142,17	2,46
BQTerrace	22	22	10471,14	307,76	140,15%	2,91	86,46%	12110,62	219,59	3,36
Time geomean			6515,11	178,18	148,33%	1,81	93,43%	6973,14	120,13	1,94
Time sum (hours)			9,84	0,27	147,06%	9,84	91,79%	10,72	0,19	10,72

Table 24: comparative encoding and decoding times with anchor values (AI HEVC 1.5x)

			Scalable Enhan Tested					Anchors		
AI HEVC SNR	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchor	Enc T [h]	Enc T/Anchor	Enc T [s]	Dec T [s]	Enc T [h]
Traffic	26	22	12905,50	311,77	173,90%	3,58	133,14%	9693,50	179,28	2,69
	30	22	12629,99	317,77	177,25%	3,51	130,29%	9693,50	179,28	2,69
PeopleOnStreet	26	22	14111,95	346,03	191,30%	3,92	134,06%	10526,45	180,89	2,92
	30	22	14104,56	347,76	192,25%	3,92	133,99%	10526,45	180,89	2,92
Kimono	26	22	5022,51	110,42	184,56%	1,40	136,86%	3669,80	59,83	1,02
	30	22	4865,66	106,74	178,41%	1,35	132,59%	3669,80	59,83	1,02
ParkScene	26	22	5757,69	145,13	175,96%	1,60	132,01%	4361,64	82,48	1,21
	30	22	5472,52	147,50	178,83%	1,52	125,47%	4361,64	82,48	1,21
Cactus	26	22	11939,00	287,17	176,85%	3,32	124,32%	9603,70	162,38	2,67
	30	22	11338,56	301,43	185,63%	3,15	118,06%	9603,70	162,38	2,67
BasketballDrive	26	22	10896,81	240,18	168,94%	3,03	123,04%	8856,15	142,17	2,46
	30	22	10585,72	244,30	171,84%	2,94	119,53%	8856,15	142,17	2,46
BQTerrace	26	22	14713,19	361,96	164,84%	4,09	121,49%	12110,62	219,59	3,36
	30	22	13349,80	362,64	165,14%	3,71	110,23%	12110,62	219,59	3,36
Time geomean			9811,59	239,18	177,36%	2,73	126,57%	7752,02	134,86	2,15
Time sum (hours)			41,03	1,01	176,83%	41,03	125,54%	32,68	0,57	32,68

Table 25: comparative encoding and decoding times with anchor values (AI HEVC SNR)

			Scalable Enhan Tested					Anchors		
RA HEVC 2x	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchors	Enc T [h]	Enc T/Anchors	Enc T [s]	Dec T [s]	Enc T [h]
Traffic	22	22	69547,52	216,15	254,75%	19,32	273,09%	25466,76	84,85	7,07
PeopleOnStreet	22	22	92752,98	280,04	227,02%	25,76	251,62%	36862,97	123,36	10,24
Kimono	22	22	32054,27	100,61	257,93%	8,90	252,60%	12689,88	39,01	3,52
ParkScene	22	22	31388,40	101,38	248,22%	8,72	267,49%	11734,35	40,84	3,26
Cactus	22	22	68278,95	199,81	239,72%	18,97	250,15%	27295,24	83,35	7,58
BasketballDrive	22	22	79301,86	213,38	243,54%	22,03	254,03%	31216,94	87,62	8,67
BQTerrace	22	22	90797,04	274,58	213,29%	25,22	251,27%	36134,62	128,74	10,04
Time geomean			61132,53	184,32	240,18%	16,98	257,04%	23783,12	76,74	6,61
Time sum (hours)			128,92	0,38	235,80%	128,92	255,85%	50,39	0,16	50,39

Table 26: comparative encoding and decoding with anchor values (RA HEVC 2x)

			Scalable Enhan Tested					RA HEVC		
RA HEVC 1.5x	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchors	Enc T [h]	Enc T/Anchors	Enc T [s]	Dec T [s]	Enc T [h]
Kimono QPB 22	22	22	36065,21	132,93	340,78%	10,02	284,20%	12689,88	39,01	3,52
ParkScene QPB 22	22	22	34189,00	137,52	336,71%	9,50	291,36%	11734,35	40,84	3,26
Cactus QPB 22	22	22	73041,20	267,84	321,34%	20,29	267,60%	27295,24	83,35	7,58
BasketballDrive QPB 22	22	22	85496,56	286,57	327,07%	23,75	273,88%	31216,94	87,62	8,67
BQTerrace QPB 22	22	22	98394,01	366,84	284,96%	27,33	272,30%	36134,62	128,74	10,04
Time geomean			59688,69	219,94	321,53%	16,58	277,73%	21491,34	68,41	5,97
Time sum (hours)			90,88	0,33	313,98%	90,88	274,78%	33,08	0,11	33,08

Table 27: comparative encoding and decoding times with anchor values (RA HEVC 1.5x)

			Scalable Enhan Tested					RA HEVC		
RA HEVC SNR	QPI Base	QPI Enhan	Enc T [s]	Dec T [s]	Dec T/Anchors	Enc T [h]	Enc T/Anchors	Enc T [s]	Dec T [s]	Enc T [h]
Traffic	26	22	88071,38	250,99	295,81%	24,46	345,83%	25466,76	84,85	7,07
	30	22	87085,34	246,32	290,30%	24,19	341,96%	25466,76	84,85	7,07
PeopleOnStreet	26	22	115185,54	319,95	259,37%	32,00	312,47%	36862,97	123,36	10,24
	30	22	114148,72	315,63	255,87%	31,71	309,66%	36862,97	123,36	10,24
Kimono	26	22	39999,62	117,88	302,20%	11,11	315,21%	12689,88	39,01	3,52
	30	22	35657,92	115,35	295,71%	9,90	280,99%	12689,88	39,01	3,52
ParkScene	26	22	38807,85	119,30	292,10%	10,78	330,72%	11734,35	40,84	3,26
	30	22	39418,39	116,69	285,72%	10,95	335,92%	11734,35	40,84	3,26
Cactus	26	22	85891,69	232,90	279,42%	23,86	314,68%	27295,24	83,35	7,58
	30	22	87005,52	229,25	275,04%	24,17	318,76%	27295,24	83,35	7,58
BasketballDrive	26	22	98682,23	254,37	290,32%	27,41	316,12%	31216,94	87,62	8,67
	30	22	98444,71	252,56	288,25%	27,35	315,36%	31216,94	87,62	8,67
BQTerrace	26	22	113562,79	328,90	255,48%	31,55	314,28%	36134,62	128,74	10,04
	30	22	110705,15	319,87	248,47%	30,75	306,37%	36134,62	128,74	10,04
Time geomean			75643,62	214,14	279,04%	21,01	318,06%	23783,12	76,74	6,61
Time sum (hours)			320,1852361	0,89443	273,92%	320,19	317,71%	100,7782	0,326532	100,78

Table 28: comparative encoding and decoding times with anchor values (RA HEVC SNR)

5.3 Description of computing platform used to determine encoding and decoding times reported in sections 5.1 and 5.2

All encodings and decodings have been performed on one core of Intel Xeon X56xx processors. Encodings have been performed on a Linux cluster made of Xeon X5670 @ 2.93 GHz.

Decodings have been tested on both Linux and Windows PC. All decodings are doable on both Linux and Windows. The decoding times come from decodings performed under Windows on a PC with a Xeon X5660 @2.8GHz processor.

Neither parallel code (for instance OMP) nor SIMD (like SSE) optimization has been used.

5.4 Expected memory usage of encoder

The expected memory usage is as follows for the **encoder.**

- AI20
  - class A+ : ~1.61 GB
  - class B : ~0.5 GB
- AI15
  - Class B: ~0.54 GB
- AI SNR
  - Class A+: ~2.16 GB
  - Class B: ~0.64 GB
- RA20
  - class A+ : ~5.98 GB
  - class B : ~1.69GB
- RA15
  - Class B: ~1.85 GB
- RA SNR
  - Class A+: ~8.18 GB
  - Class B: ~2.26 GB

5.5 Expected memory usage of decoder

The expected memory usage is as follows for the **decoder.**

- AI20
  - class A+ : ~0.98 GB
  - class B : ~0.30 GB
- AI15
  - Class B: ~0.32 GB
- AI SNR
  - Class A+: ~1.17 GB
  - Class B: ~0.35 GB
- RA20
  - class A+ : ~3.38 GB
  - class B : ~0.95 GB
- RA15
  - Class B: ~1.03 GB
- RA SNR
  - Class A+: ~4.47 GB
  - Class B: ~1.23 GB

5.6 Complexity characteristics of encoder motion estimation and partitioning selection in enhancement layer(s)

The motion estimation and the partitioning selection involved in the CSC enhancement layer coding are identical to those of the HM6.1, with the same number of reference pictures.
No AMP is used. Maximum Coding Unit size is equal to 64x64.

5.7 Complexity characteristics of decoder motion compensation in enhancement layer(s)

Motion compensation performed in the enhancement layer is similar to that of the HM6.1, except an additional motion compensation step is invoked during the Base Mode prediction picture computation. This is explained in section 2.4.1.2.4.

5.8 Complexity characteristics of encoder intra-frame prediction type and partitioning selection in enhancement layer(s)

The intra-frame prediction partitioning selection in enhancement pictures consists in iterative process that aims at improving the segmentation of the intra enhancement picture into transform blocks, with associated sizes and types (see section 2.3.9). During this iterative competition mechanism, rate and distortion values are computed for considered block types. The distortion values are computed as the L2 distortion between original transform block and de-quantized block. The rate values computation does not involve any effective entropy coding process. Instead, it is estimated as a function of the probabilities associated to each quantum involved in the considered block.

With respect to intra-frame prediction, only one prediction mode if employed in Intra enhancement pictures, i.e. inter-layer intra prediction. No spatial prediction takes place in Intra pictures. Therefore, no rate distortion optimized coding mode selection is invoked, as in classical video standard.

5.9 Complexity characteristics of decoder intra-frame prediction operation in enhancement layer(s)

No intra frame prediction operation is done on the decoder side. In the same way as on the encoder side, the whole enhancement Intra picture is predicted from the underlying base picture, optionally up-sampled in case of spatial scalability.

5.10 Complexity characteristics of upsampling filters and transforms specific in enhancement layer(s)

The DCT-IF interpolation filters used in HEVC for motion compensation are re-used for up-sampling matters, with adapted phases (see section 2.2). This takes the form of:

- 8-tap interpolation filters for luma
- 4-tap interpolation filters for chroma

With respect to the computation of the base mode prediction picture, a bi-linear filter is employed to up-sample the base picture's temporal residual data.

5.11 Complexity characteristics of quantization and inverse quantization in enhancement layer(s)

With respect to Intra enhancement pictures, the quantization consists, given a DCT sample to quantize and the optimal quantizer used for the considered DCT channel, in determining the quantization interval the DCT sample belongs to. No calculation is needed; everything is tabulated at this stage.

Inverse quantization consists in providing a considered quantized DCT sample with the centroid value of the interval the quantized sample belongs to. Again, no arithmetic operation is needed here.

With respect to Inter enhancement pictures, the HM6.1 quantization and inverse quantization are re-used without any modification.

5.12 Complexity characteristics of encoder entropy coding operation in enhancement layer(s)

The entropy coding in Intra picture consists in a context-free, non-adaptive arithmetic coder. No context is needed because the probability of occurrence of each quantum is known a priori thanks to the knowledge of the Generalized Gaussian Distribution. These probabilities are computed off-line and stored within the data associated to each quantizer.
Context free coding also allows a straightforward design of the codec with the so-called "spatial random access" feature.

With respect to Inter enhancement pictures, the entropy coding of the HM6.1 is re-used as is. A limited number of syntax elements are added to the HEVC specification. Syntax elements added in the slice header syntax are unary coded elements. Coding Unit level new syntax elements are coded with the CABAC entropy encoder. For each of them, context models with minimal size 1 are employed.

5.13 Complexity characteristics of decoder entropy decoding operation in enhancement layer(s)

The entropy decoding process in Intra enhancement picture consists in a context-free, non-adaptive arithmetic decoding process. In the same way as on the encoder side, fixed probabilities, issued from the knowledge of the Generalized Gaussian Distribution, are considered.

Concerning Inter enhancement pictures, the CABAC entropy decoder is used in the same way as in the base layer.

5.14 Degree of capability for parallel processing

By construction, the newly introduced coding scheme for Intra frames can handle easily a high level of parallelization because

- there is no intra block prediction,
- the entropy coder is a context-free coder.

This allows the segmentation of Intra frames into many slices with no impact on the compression performance.

Concerning the Inter frames, the core HEVC capability for parallel processing remains intact as no newly introduced Inter tool should impact this capability negatively.

6 Software implementation description

The CSC codec has been written in C++ and uses the source code of HM6.1. The preliminary scalable framework has been developed by Vidyo and Canon has implemented the different coding tools presented in this document. Neither parallel code (for instance OMP) nor SIMD (like SSE) optimization has been used.

7 Closing remarks

This document presents the video coding tools of the Canon Scalable Codec (CSC) as well as the coding performance in response to the Joint Call for Proposals for the scalable video coding extensions of HEVC.

In terms of coding efficiency, it is reported that the proposed codec shows scalable performance with a mean overhead of about 8.5% in All Intra configurations, 6.7% in Random Access spatial configurations and 3.1% in random Access SNR configuration compared to the monocast (single layer) HM6.1. Overall, a gain of 44.3% over the simulcast enhancement layer is obtained with the proposed scalability layer coding system. This corresponds to a gain of roughly 29% compared to the simulcast HM6.1, in the coding of the base plus the enhancement layer.

The CSC software is based on the source code of HM6.1 initially modified by Vidyo to provide a scalable extension framework to HEVC [4, 17]. The classical structure of the software allows easy integration and test of new tools. The CSC bitstream is totally compatible with HM6.1. The source code could thus be used as a basis for the collaborative design phase.

8 Acknowledgment

Vidyo graciously provided to Canon their preliminary scalable codec based on HEVC described in Vidyo contributions JCTVC-F290 and JCTVC-G078. Parts of the Vidyo software have been used in Canon response to the Joint Call for Proposals on Scalable Video Coding Extensions of HEVC described in this contribution.

9 References

[1] "High efficiency video coding (HEVC) text specification draft 6", JCTVC-H1003, November 2011, Geneva.

[2] "Joint Call For Proposals on Scalable Video Coding Extensions of High Efficiency Video Coding (HEVC)", ISO/IEC JTC 1/SC 29/WG 11 (MPEG) Doc. N12957, July 2012, Stockholm, Sweden.

[3] ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services," March, 2010.

[4] "Scalability support in HEVC", JCTVC-F290, D. Hong, W.Jang, J.Boyce, Torino, IT, July 2011.

[5] "Scalable structures and inter-layer prediction for HEVC scalable extension", H.M. Choi, J.Nam, D.Sim, JCTVC-F096, Torino, IT, July 2011.

[6] "Entropy Constrained Vector Quantization, IEEE Trans. on acoustics, speech and signal processing", P. Chou, T. Lookabaugh, R. Gray, September, vol. 37(1), January 1989.

[7] "Low Complexity Scalable Extension of HEVC intra pictures", S.Lasserre, F. Le Léannec, E. Nassor, JCTVC-G248, November 2011.

[8] "Source Coding: Part I of Fundamentals of Source and Video Coding", T.Wiegand, H.Schwarz.

[9] http://en.wikipedia.org/wiki/Mathematical_morphology.

[10] "Image Analysis and Mathematical Morphology", Vol. 1, Jean Serra.

[11] "Encoding with fixed Lagrange multipliers", Bin Li, Li Li, Jinlei Zhang, Jizheng Xu, Houqiang Li, JCTVC-J042, July 2012, Stockholm, Sweden.

[12] "HM6: High Efficiency Video Coding (HEVC) Test Model 6 Encoder Description", JCTVC-H1002, San Jose, February 2012.

[13] "Modified SAO edge offsets", K. Andersson, P. Wennersten, R. Sjöberg, JCTVC-G490, 21-30 November, 2011, Geneva, Switzerland.

[14] "On additional SAO Band Offset classifications", G. Laroche, T. Poirier, P. Onno, JCTVC-G246, 21-30 November, 2011, Geneva, Switzerland.

[15] "CE8.a.4: One-stage/Two-stage SAO and ALF with LCU-based syntax", C.-Y. Chen, C.-Y. Tsai, C.-M. Fu, Y.-W. Huang, S. Lei, I. S. Chong, M. Karczewicz, T. Yamakage, T. Itoh, T. Watanabe, T. Chujoh, JCTVC-H0274, San Jose, February 2012.

[16] "CE8.b: ALF with single filter type", M. Budagavi, JCTVC-H0068, San Jose, February 2012.

[17] "Information for HEVC scalability extension", JCTVC-G078, J.Boyce, D. Hong, W.Jang, A. Abbas, Geneva, CH, November 2011.

10 Patent rights declaration(s)

Canon Research Centre France may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).