Final Report

Coding Efficiency and Computational Complexity of Video Coding Standards-Including High Efficiency Video Coding (HEVC)

Spring 2014

Multimedia Processing EE 5359

Advisor: Dr. K. R. Rao

Department of Electrical Engineering

University of Texas, Arlington

Zarna Patel

1001015672

zarnaben.patel@mavs.uta.edu

ACRONYMS AND ABBREVIATIONS

- AMVP: Advanced Motion Vector Prediction
- AVC: Advanced Video Coding
- CABAC: Context Adaptive Binary Arithmetic Coding
- CAVLC: Context Adaptive Variable Length Coding
- CHC: Conversational High Compression
- COD: Coded macroblock indication
- CTB: Coding Tree Block
- CTU: Coding Tree Unit
- CB: Coding Block
- CU: Coding Unit
- DCT: Discrete Cosine Transform
- DBF: Deblocking Filter
- DSP: Digital Signal Processor
- DST: Discrete Sine Transform
- GOB: Group of Blocks
- HD: High Definition
- HEVC: High Efficiency Video Coding
- HLP: High Latency profile
- HP: High Profile
- JCT-VC: Joint Collaborative Team on Video Coding
- MB: Macroblock
- MSE: Mean square Error
- MV: Motion Vector
- NAL: Network Abstraction Layer
- PB: Prediction Block
- PSNR: Peak signal-to-noise ratio
- PU: Prediction Unit
- RPL: Reference Picture List
- SAO: Sample Adaptive Offset
- SP: Spatial (intra) Prediction
- SVC: Scalable Video Coding
- TB: Transform Block
- TMVP: Temporal Motion Vector Prediction
- TS: Transform Skip
- TU: Transform Unit
- URQ: Uniform Reconstruction Quantization
- VCL: Video Coded Layer
- VGA: Video Graphics Array
- WPP: Wavefront Parallel Processing

I. INTRODUCTION:

The primary goal of digital video coding standards has been to optimize coding efficiency. Coding efficiency is the ability to minimize the bit rate necessary for representation of video content to reach a given level of video quality or, as alternatively formulated, to maximize the video quality achievable within a given available bit rate [24]. High Efficiency Video Coding (HEVC) is a new video coding standard developed by the JCT-VC group within ISO/IEC and ITU-T [8]. An increasing variety of services, the growing popularity of HD video, and the formats going beyond HD (e.g., 4K × 2K or 8K × 4K resolution) are creating even stronger needs for compression capabilities superior to H.264/AVC [14]. Now mobile devices and tablet personal computers will also need to receive and display HD video. HEVC has been designed to address essentially all existing applications of previous standards and to particularly focus on two key issues: increased video resolution and the increased use of parallel processing architectures. The syntax of HEVC is generic, and its design elements can also be attractive for other application domains that have not been used by the previous standards [16]. HEVC is targeted to provide the same quality as H.264 [14] at about half the bit-rate and will replace soon its predecessor in multimedia consumer applications [1]. This increasing efficiency has supported the evolution of multimedia applications towards higher spatial and temporal resolution formats (e.g. HD) and the arising of more complex applications, such as 3D video [1].

As expected, higher coding efficiency is obtained at the expense of a significant increase in computational complexity, mainly resulting from much more intensive processing requirements, nested data structures, and optimization algorithms dealing with larger amounts of data. The problem of the high computational complexity required for achieving efficient video encoding directly affects the development of cost-effective multimedia systems; thus, it should be closely connected to the implementation of standards. Like previous video coding standards, HEVC incorporates a significant number of different coding tools, most of them using different parameters, to which several values can be assigned. Therefore, its overall complexity is actually the result of a cumulative contribution of all these coding tools and parameters. Different combinations of such tools and parameters give rise to specific encoding configurations, which necessarily result in quite different performances and complexities [26].

I. Objective:

The objective of this project is to analyze the coding efficiency and computational complexity that can be achieved by use of the emerging High Efficiency Video Coding (HEVC) standard, relative to the coding efficiency characteristics of its major predecessors including, H.263 [29], and H.264/MPEG-4 Advanced Video Coding (AVC) [14]. The compression capabilities of several generations of video coding standards are compared by means of peak signal-to-noise ratio (PSNR). In a previous work, an implementation based on HM9.0 reference software was presented [6], but in this project, HM13.0 reference software will be used [7].

II. HEVC Encoder & Decoder:

HEVC standard is based on the same motion-compensated hybrid coding as its predecessors, from H.261 to H.264 [15]. The new standard is not a revolutionary design; instead, it has a lot of small improvements that, when put together, lead to a considerable bit-rate reduction. The tests performed during the standardization process show that HEVC may compress at half the bit-rate of H.264 with the same quality [8], at the expense of a higher complexity.

Fig. 1 [9] depicts the block-diagram of a hybrid HEVC video encoder. In the following, the various features involved in hybrid video coding using HEVC are highlighted:

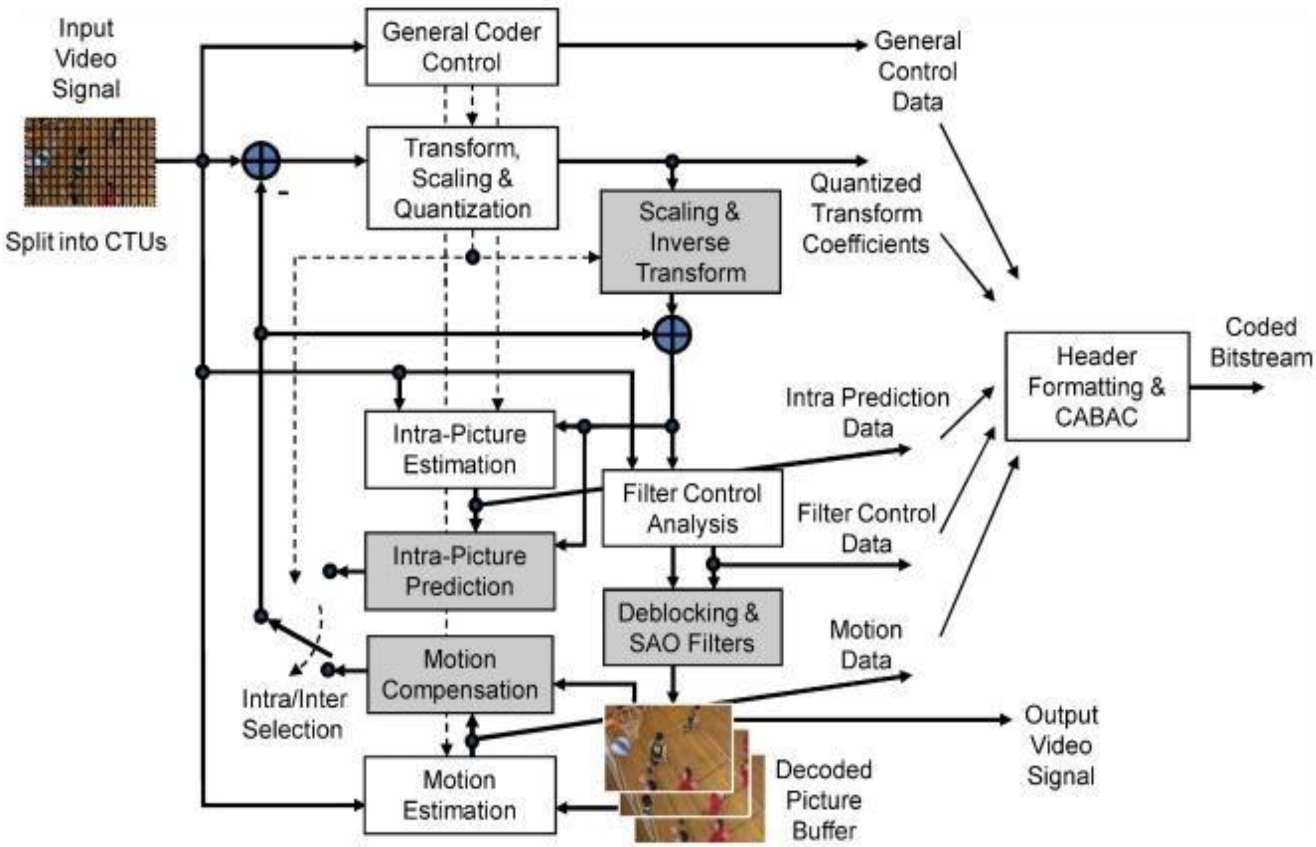


Fig. 1. Block diagram of video encoder for HEVC [9].

1) Coding tree units and coding tree block (CTB) structure:

The core of the coding layer in previous standards was the macroblock, containing a 16×16 block of luma samples and, in the usual case of 4:2:0 color sampling, two corresponding 8×8 blocks of chroma samples; whereas the analogous structure in HEVC is the coding tree unit (CTU), which has a size selected by the encoder and can be larger than a traditional macroblock. The CTU consists of a luma CTB and the corresponding chroma CTBs and syntax elements. The size L×L of a luma CTB can be chosen as L = 16, 32, or 64 samples, with the larger sizes typically enabling better compression. The nominal vertical and horizontal relative locations of luma and chroma samples in pictures are shown in Fig. 2 [34].

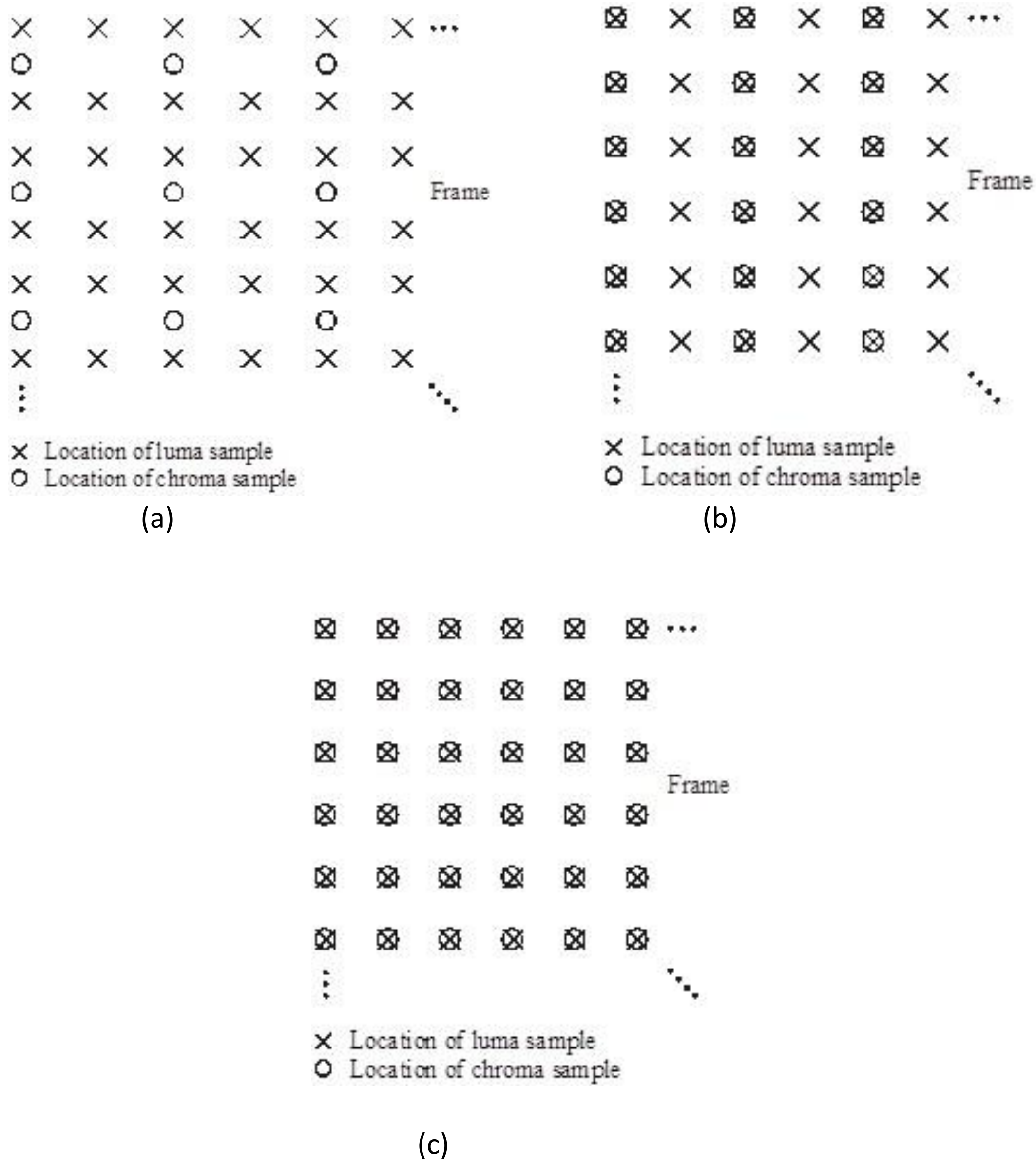
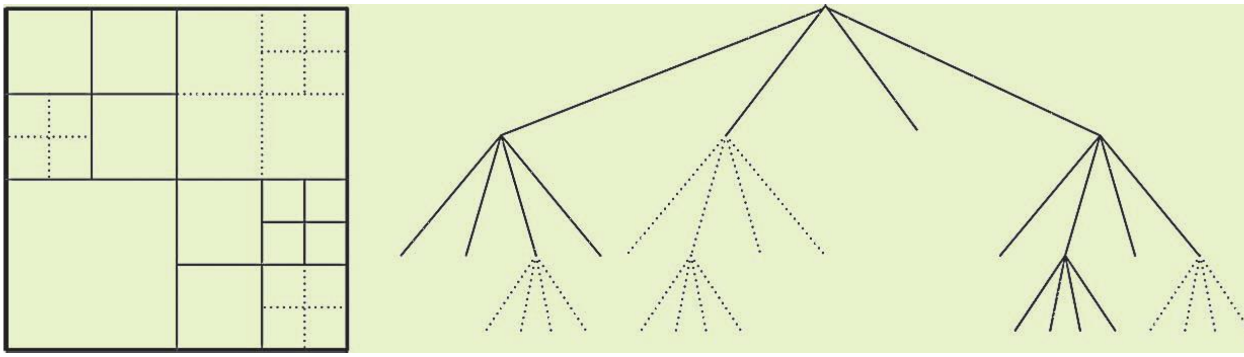


Fig. 2. Nominal vertical and horizontal locations of luma and chroma samples in a picture (a) 4:2:0 (b) 4:2:2 (c) 4:4:4 [34].



(a) (b)

Fig. 3. Subdivision of a 64 × 64 luma CTB into CBs and TBs. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. (a) The CTB with its partitioning. (b) The corresponding quadtree. In this example, the leaf nodes are each 8 × 8 in size, although, in general, a TB can actually be as small as 4 × 4 [16].

2) Coding units (CUs) and coding blocks (CBs):

CTBs are then partitioned into coding blocks (CBs), signaled via a quadtree structure, as illustrated in Fig. 3 [16]. The quadtree syntax of the CTU specifies the size and positions of its luma and chroma CBs. The root of the quadtree is associated with the CTU. Hence, the size of the luma CTB is the largest supported size for a luma CB. The splitting of a CTU into luma and chroma CBs is signaled jointly. One luma CB and ordinarily two chroma CBs, together with associated syntax, form a coding unit (CU) valid for 4:2:0 format. A CTB may contain only one CU or may be split to form multiple CUs, and each CU has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs).

3) Prediction units (PUs) and prediction blocks (PBs):

Each CU can be further split into smaller units, which form the basis for prediction. These units are called PUs. Each CU may contain one or more PUs, and each PU can be as large as its root CU or as small as 4 × 4 in luma block sizes. While an LCU (Largest Coding Unit) can recursively split into smaller and smaller CUs, the splitting of a CU into PUs is nonrecursive (it can be done only once). PUs can be symmetric or asymmetric. Symmetric PUs can be square or rectangular (nonsquare) and are used in both intraprediction (uses only square PUs) and interprediction. In particular, a CU of size 2N × 2N can be split into two symmetric PUs of size N × 2N or 2N × N or four PUs of size N × N. Asymmetric PUs are used only for interprediction. This allows partitioning, which matches the boundaries of the objects in the picture. Fig. 4 [8] shows partitioning of a CU to symmetric and asymmetric PUs.

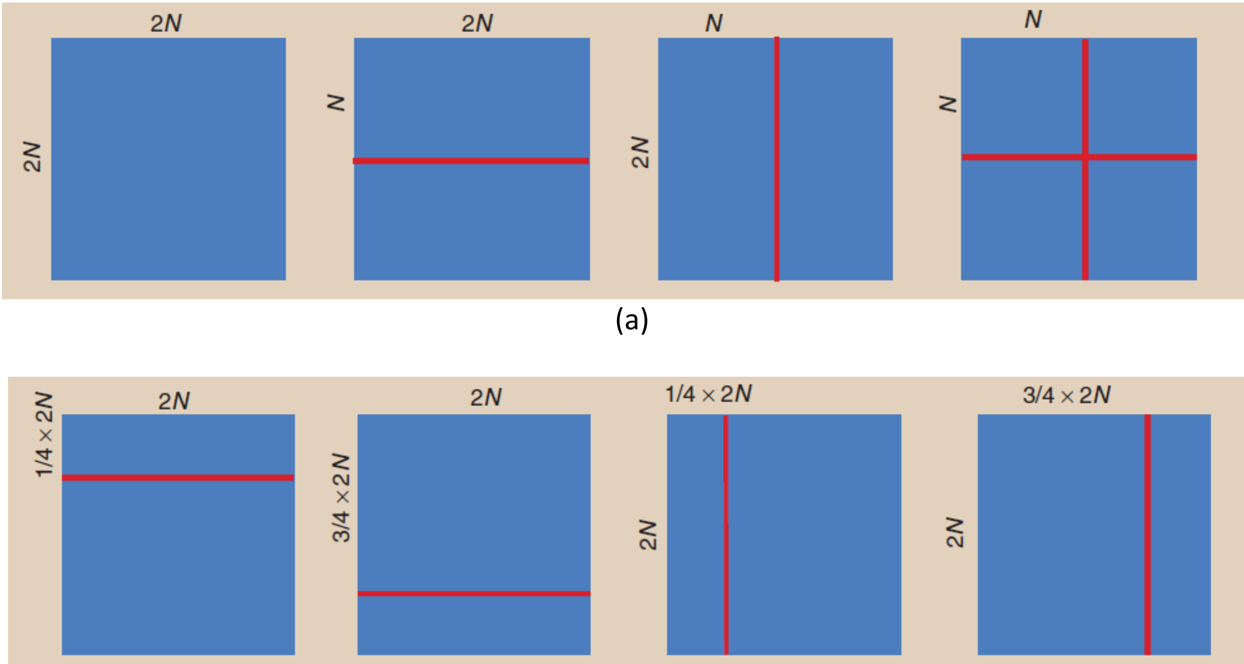


Fig. 4. (a) Symmetric and (b) asymmetric PUs [8].

4) Transform Units and Transform blocks:

The prediction residual is coded using block transforms. A TU tree structure has its root at the CU level. The luma CB residual may be identical to the luma transform block (TB) or may be further split into smaller luma TBs. The same applies to the Chroma TBs. Integer basis functions similar to those of a discrete cosine transform (DCT) are defined for the square TB sizes 4×4, 8×8, 16×16, and 32×32. For the 4×4 transform of luma intrapicture prediction residuals, an integer transform derived from a form of discrete sine transform (DST) is alternatively specified. Fig. 5 [8] illustrates an example for partitioning a 32 × 32 CU into PUs and TUs.

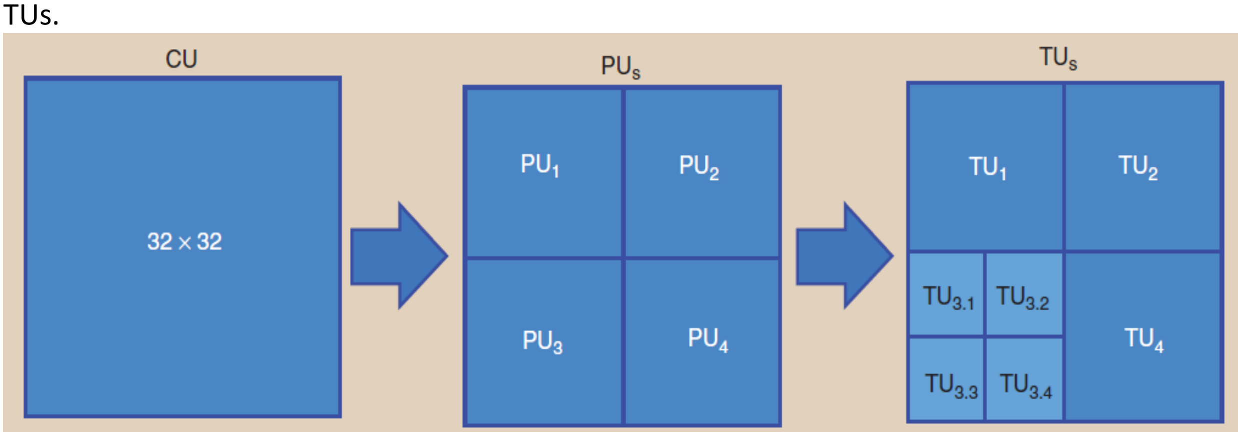


Fig. 5. An example of partitioning a 32×32 CU to PUs and TUs [8].

5) Motion vector signaling:

Advanced motion vector prediction (AMVP) is used, including derivation of several most probable candidates based on data from adjacent PBs and the reference picture. A merge mode for MV coding can also be used, allowing the inheritance of MVs from temporally or spatially neighboring PBs. Moreover, compared to H.264/MPEG-4 AVC, improved skipped and direct motion inference are also specified.

6) Motion compensation:

Unlike a two-stage interpolation process adopted in H.264, HEVC uses separable 8-tap filter for ½ pixels and 7-tap filter for ¼ pixels (Table 1) [9]. Integer ($A_{i,j}$) and fractional pixel positions (lower case letters) for luma interpolation are shown in Fig. 6 [9]. Similarly 4-tap filter coefficients for chroma fractional (1/8 accuracy) pixel interpolation are listed in Table 2 [9].

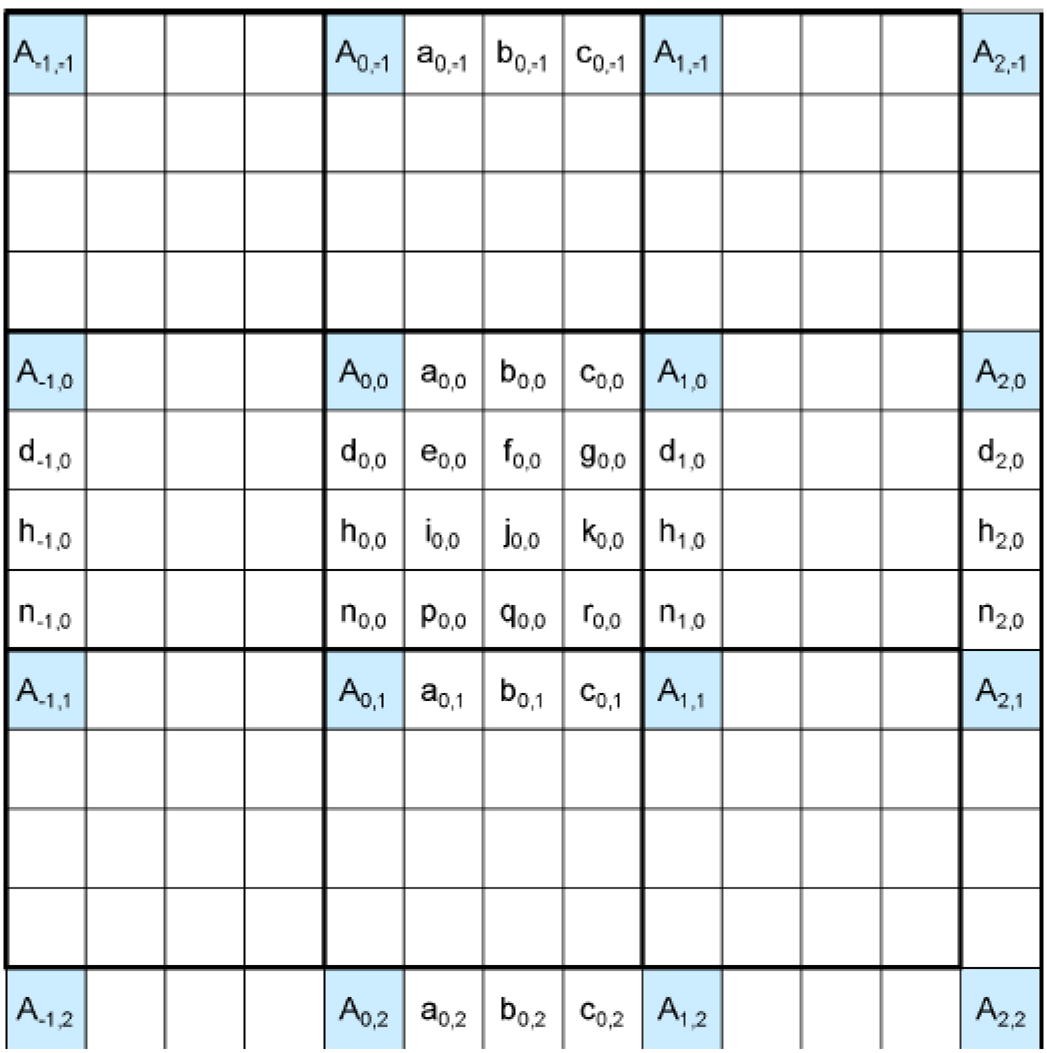


Fig. 6. Integer and fractional sample positions for luma interpolation [9]

Index i	-3	-2	-1	0	1	2	3	4
hfilter[i]	-1	4	-11	40	40	-11	4	1
qfilter[i]	-1	4	-10	58	17	-5	1	

Table 1. Filter coefficients for luma fractional sample interpolation [9]

The samples labeled a0,j, b0,j, c0,j, d0,0, h0,0, and n0,0 are derived from the samples Ai,j by applying the eight-tap filter for half-sample positions and the seven-tap filter for the quarter-sample positions as follows:

$$\mathbf{a0,j = (\textstyle\sum_{i=-3..3} Ai,j\ qfilter[i]) >> (B - 8)}$$
$$\mathbf{b0,j = (\textstyle\sum_{i=-3..4} Ai,j\ hfilter[i]) >> (B - 8)}$$
$$\mathbf{c0,j = (\textstyle\sum_{i=-2..4} Ai,j\ qfilter[1 - i]) >> (B - 8)}$$
$$\mathbf{d0,0 = (\textstyle\sum_{i=-3..3} A0,j\ qfilter[j]) >> (B - 8)}$$
$$\mathbf{h0,0 = (\textstyle\sum_{i=-3..4} A0,j\ hfilter[j]) >> (B - 8)}$$
$$\mathbf{n0,0 = (\textstyle\sum_{j=-2..4} A0,j\ qfilter[1 - j]) >> (B - 8)}$$

where the constant B ≥ 8 is the bit depth of the reference samples (and typically B = 8 for most applications) and the filter coefficient values are given in Table 2 [9]. In these formulae, >> denotes an arithmetic right shift operation.

The samples labeled e0,0, f0,0, g0,0, i0,0, j0,0, k0,0, p0,0, q0,0, and r0,0 can be derived by applying the corresponding filters to samples located vertically adjacent a0,j, b0,j and c0,j positions as follows:

$$\mathbf{e0,0 = (\textstyle\sum_{v=-3..3} a0,v\ qfilter[v]) >> 6}$$
$$\mathbf{f0,0 = (\textstyle\sum_{v=-3..3} b0,v\ qfilter[v]) >> 6}$$
$$\mathbf{g0,0 = (\textstyle\sum_{v=-3..3} c0,v\ qfilter[v]) >> 6}$$
$$\mathbf{i0,0 = (\textstyle\sum_{v=-3..4} a0,v\ hfilter[v]) >> 6}$$
$$\mathbf{j0,0 = (\textstyle\sum_{v=-3..4} b0,v\ hfilter[v]) >> 6}$$
$$\mathbf{k0,0 = (\textstyle\sum_{v=-3..4} c0,v\ hfilter[v]) >> 6}$$
$$\mathbf{p0,0 = (\textstyle\sum_{v=-2..4} a0,v\ qfilter[1 - v]) >> 6}$$
$$\mathbf{q0,0 = (\textstyle\sum_{v=-2..4} b0,v\ qfilter[1 - v]) >> 6}$$
$$\mathbf{r0,0 = (\textstyle\sum_{v=-2..4} c0,v\ qfilter[1 - v]) >> 6}$$

The interpolation filtering is separable when B is equal to 8, so the same values can be computed in this case by applying the vertical filtering before the horizontal filtering. When implemented appropriately, the motion compensation process of HEVC can be performed using only 16 blocks storage elements.

Index	−1	0	1	2
filter1[i]	−2	58	10	−2
filter2[i]	−4	54	16	−2
filter3[i]	−6	46	28	−4
filter4[i]	−4	36	36	−4

Table 2. Filter coefficients for chroma fractional pels [9]

The fractional sample interpolation process for the chroma components is similar to the one for the luma component, except that the number of filter taps is 4 and the fractional accuracy is 1/8 for the usual 4:2:0 chroma format case. HEVC defines a set of four-tap filters for eighth sample positions, as given in Table 2 for the case of 4:2:0 chroma format.

Filter coefficient values denoted as filter1[i], filter2[i], filter3[i], and filter4[i] with i = −1, 0 1, 2 are used for interpolating the 1/8th, 2/8th, 3/8th, and 4/8th fractional positions for the chroma samples, respectively. Using symmetry for the $5/8^{th}$, $6/8^{th}$, and $7/8^{th}$ fractional positions, the mirrored values of filter3[1−i], filter2[1−i], and filter1[1−i] with i = −1, 0, 1, 2 are used, respectively.

7) Intra prediction:

HEVC has 35 luma intra prediction modes, including DC and planar modes [8]. For intra decoded CUs, a prediction is calculated based on the boundary pixels belonging to previously decoded neighboring CUs. It is the same type of intra prediction used in H.264 with more directional modes shown in Fig. 7 [9]. The number of supported prediction modes varies based on the PU size shown in Table 3 [8]. HEVC includes a planar intra prediction mode, which is useful for predicting smooth picture regions. In planar mode, the prediction is generated from the average of two linear interpolations (horizontal and vertical) [21].

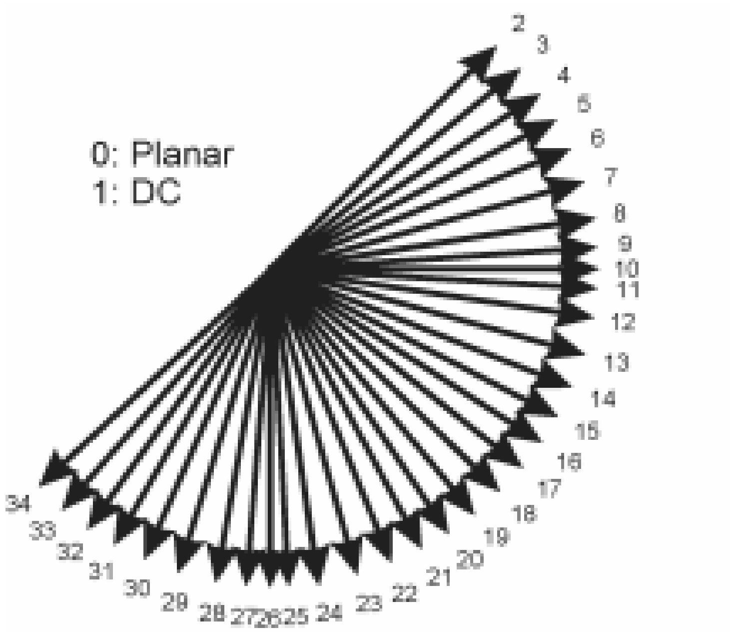


Fig. 7. Intra Prediction Modes for HEVC [9]

PU Size	Intraprediction Modes
4 × 4	0–16, 34
8 × 8	0–34
16 × 16	0–34
32 × 32	0–34
64 × 64	0–2, 34

Table 3. Luma intraprediction modes supported for different PU sizes [8]

To further improve the intra prediction efficiency of the chroma components, besides the increasing prediction directions, HEVC introduces a new intra chroma prediction mode which utilizes the correlation between chroma and luma samples [22]. Chroma samples are predicted from the reconstructed luma samples around the prediction block by modelling chroma samples as a linear function of luma samples. The model parameters are determined by linear regression using the neighboring reconstructed pixels of the current coding luma and chroma blocks.

8) Inter prediction:

For inter decoded CUs, a prediction is calculated based on the previously decoded pictures that are stored in the reference frames buffer. Inter prediction is carried out at

Prediction Blocks (PBs) basis. PBs may meet their corresponding CBs in size or may be further split into PBs. Like in H.264 [14], [15], unidirectional, bidirectional and weighted prediction may be used.

9) Quantization control:

As in H.264/MPEG-4 AVC [14], uniform reconstruction quantization (URQ) is used in HEVC, with quantization scaling matrices supported for the various transform block sizes.

10) Entropy coding:

Context adaptive binary arithmetic coding (CABAC) is used for entropy coding. This is similar to the CABAC scheme in H.264/MPEG-4 AVC [14], but has undergone several changes to improve its throughput speed (especially for parallel-processing architectures) and its compression performance, and to reduce its context memory requirements.

11) Deblocking filter (DBF):

It is the same type of DBF used in H.264. An 8×8 grid is used instead of 4×4 in order to reduce the complexity. DBF uses a 8x8 sample grid since it causes no noticeable degradation and significantly improves parallel processing because the DBF no longer causes cascading interactions with other operations. Another change is that HEVC only allows for three DBF strengths of 0 to 2. HEVC also requires that the DBF first apply horizontal filtering for vertical edges to the picture and only after that it applies vertical filtering for horizontal edges to the picture. This allows for multiple parallel threads to be used for the DBF.

12) Sample Adaptive Offset (SAO):

After the DBF, the reconstructed pels are classified into different categories. An offset is added to each pel based on its category in order to improve the quality of the final reconstructed pictures.

III. <u>HEVC Profiles, Tiers and Levels:</u>

In HEVC, conformance points are defined by **profile** (combinations of coding tools)**, levels** (picture sizes, maximum bit rates etc.) and **tiers** (for bit rate and buffering capability). A conforming bitstream must be decodable by any decoder that is conforming to the given profile/tier/level combination. Three profiles have been defined [9]:

(1) **“Main” profile:** Only 8-bit video with YC_bC_r 4:2:0 is supported. Wavefront processing can only be used when multiple tiles in a picture are not used.

(2) **“Main Still Picture” profile:** It is used for still-image coding applications. Bitstream contains only a single (intra) picture, and it includes all (intra) coding features of Main profile

(3) **“Main 10” profile:** It additionally supports up to 10 bits per sample, and also includes all coding features of Main profile

The HEVC standard defines two tiers, **Main** and **High**, and thirteen levels. These 13 levels cover all important picture sizes ranging from VGA at low end up to 8K x 4K at high end. Tiers and levels with maximum property values are shown in Table 4. For levels below level 4 only the Main tier is allowed [9][17]. The Main tier is a lower tier than the High tier. The Main tier was designed for most applications while the High tier was designed for very demanding applications.

Level	Max luma sample rate (samples/s)	Max luma picture size (samples)	Max bit rate for Main and Main 10 profiles (kbit/s)		Example picture resolution @ highest frame rate
			Main tier	High tier	
1	552,960	36,864	128	–	176x144@15.0
2	3,686,400	122,880	1,500	–	352x288@30.0
2.1	7,372,800	245,760	3,000	–	640x360@30.0
3	16,588,800	552,960	6,000	–	960x540@30.0
3.1	33,177,600	983,040	10,000	–	1280x720@33.7
4	66,846,720	2,228,224	12,000	30,000	2,048x1,080@30.0
4.1	133,693,440		20,000	50,000	2,048x1,080@60.0
5	267,386,880	8,912,896	25,000	100,000	4,096x2,160@30.0
5.1	534,773,760		40,000	160,000	4,096x2,160@60.0
5.2	1,069,547,520		60,000	240,000	4,096x2,160@120.0
6	1,069,547,520	35,651,584	60,000	240,000	8,192x4,320@30.0
6.1	2,139,095,040		120,000	480,000	8,192x4,320@60.0
6.2	4,278,190,080		240,000	800,000	8,192x4,320@120.0

Table 4. Tiers and levels with maximum property values [17]

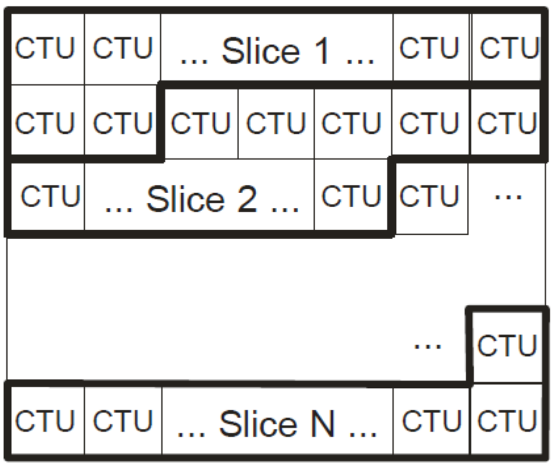
IV. HEVC – High-layer syntax structure:

The high-level syntax structure of HEVC is similar to that of H.264 [9]. The two layer structures (Network Abstraction Layer-NAL and Video Coded Layer-VCL) have been kept. Parameter sets contain information that can be shared for the decoding of several pictures or regions of the decoded video. The parameter set structure provides a robust mechanism for conveying data that are essential to the decoding process. Each syntax structure is placed into a logical data packet called a network abstraction layer (NAL) unit.

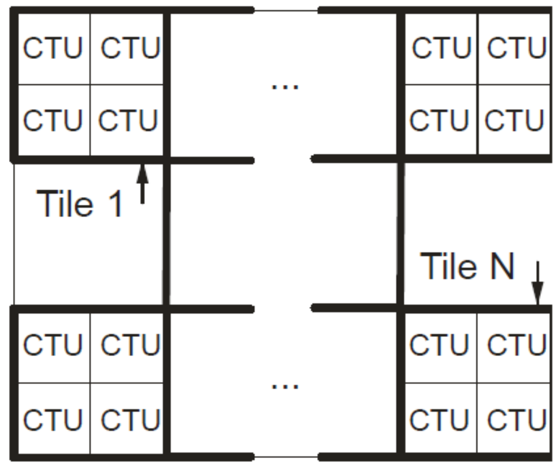
In the VCL, the pictures are divided into Coding Tree Units (CTUs), each one of them consisting of one luma and two chroma Coding Tree Blocks (CTBs). Luma CTBs size may be up to 64×64 pels. Chroma CTBs size may be up to 32×32 pels when 4:2:0 sampling is used. CTBs may be directly encoded or quadtree split into multiple CBs (Coding Blocks). Luma CBs size may be as small as 8×8 pels.

V. HEVC- Slices, Tiles and Wavefronts:

A slice is a series of CTUs that can be decoded independently from other slices of the same picture (except for in-loop filtering of the edges of the slice). A slice can either be an entire picture or a region of a picture. One of the main purposes of slices is resynchronization after data losses. An example partitioning of a picture into a slice structure is shown in Fig. 8(a) [16]. To enable parallel processing and localized access to picture regions, the encoder can partition a picture into rectangular regions called tiles. Fig. 8(b) [16] shows an example. Tiles are also independently decodable but can share some header information when multiple tiles are used within a slice.



(a)



(b)

Fig. 8. Subdivision of a picture into (a) slices and (b) tiles [16].

An additional supported form of enabling parallelism is for the encoder to use wavefront parallel processing (WPP), in which a slice is divided into rows of CTUs. With WPP, the encoding or decoding of CTUs of each row can begin after processing only two of the CTUs of the preceding row, thus enabling different processing threads to work on different rows of the picture at the same time, as shown in Fig. 9 [16]. (To minimize the difficulty of implementing decoders, encoders are prohibited from using WPP when using multiple tiles per picture.)

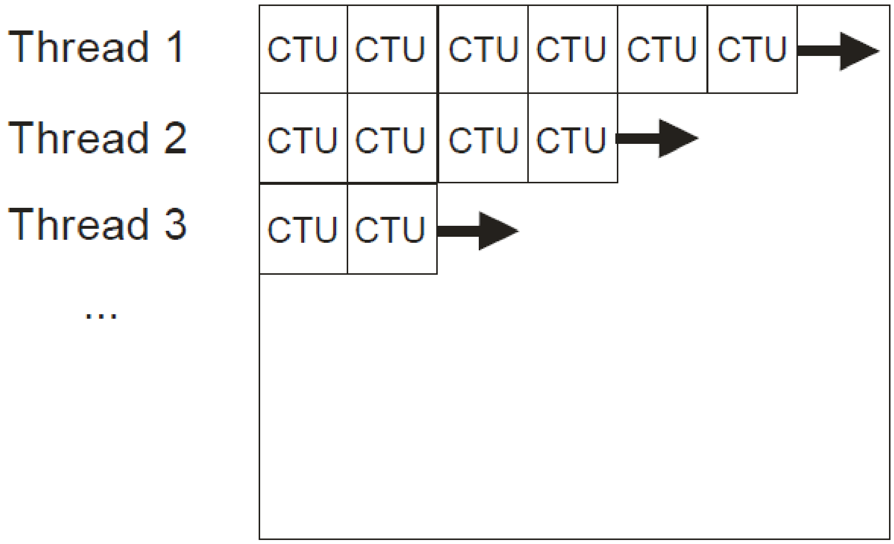


Fig. 9. wavefront parallel processing [16].

VI. ITU-T Recommendation H.263:

The first version of ITU-T Rec. H.263 [29] defines syntax features that are very similar to those of H.262/MPEG-2 [30] Video, but it includes some changes that make it more efficient for low-delay low bit-rate coding. The coding of motion vectors has been improved by using the component-wise median of the motion vectors of three neighboring previously decoded blocks as the motion vector predictor. The candidate predictors for the differential coding are taken from three surrounding macroblocks as indicated in Fig. 10 [29]. The predictors are calculated separately for the horizontal and vertical components.

In the special cases at the borders of the current GOB or picture, the following decision rules are applied in increasing order:

1) When the corresponding macroblock was coded in INTRA mode (if not in PB-frames mode) or was not coded (COD = 1), the candidate predictor is set to zero.

2) The candidate predictor MV1 is set to zero if the corresponding macroblock is outside the picture (at the left side).

3) Then, the candidate predictors MV2 and MV3 are set to MV1 if the corresponding macroblocks are outside the picture (at the top) or outside the GOB (at the top) if the GOB header of the current GOB is non-empty.

4) Then, the candidate predictor MV3 is set to zero if the corresponding macroblock is outside the picture (at the right side).

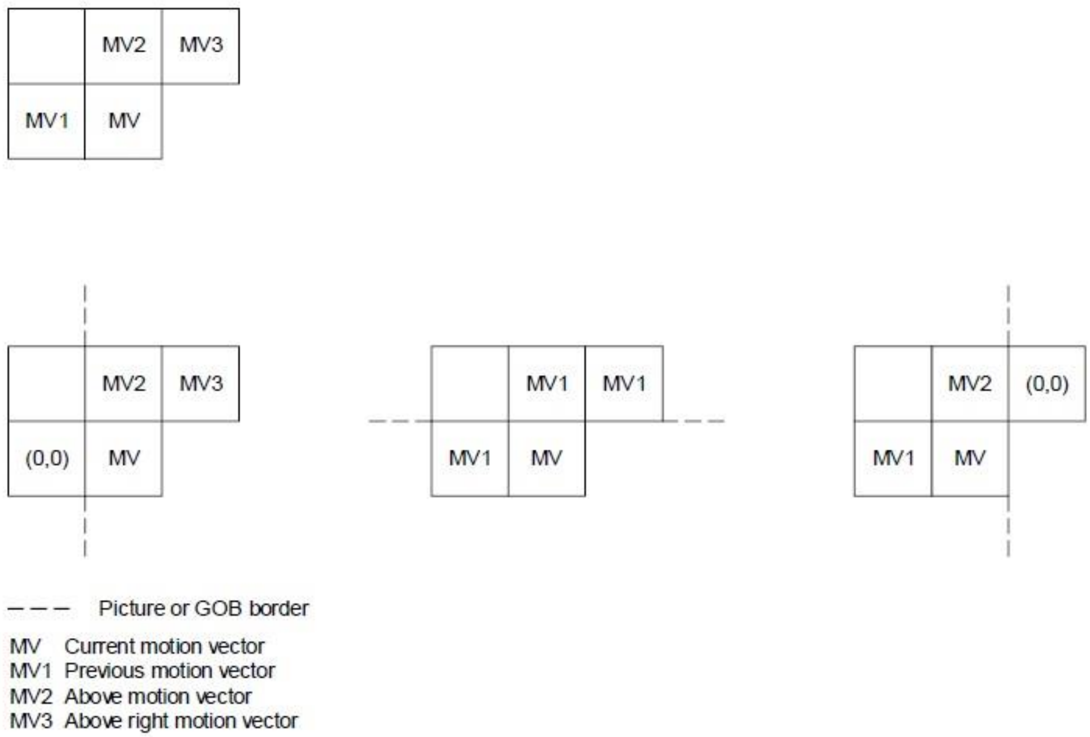


Fig. 10. Motion vector prediction [29].

The transform coefficient levels are coded using a 3-D run-level-last VLC, with tables optimized for lower bit rates. The first version of H.263 contains four annexes (annexes D through G) that specify additional coding options, among which annexes D and F are frequently used for improving coding efficiency. The usage of annex D allows motion vectors to point outside the reference picture, a key feature that is not permitted in H.262/MPEG-2 Video. Annex F introduces a coding mode for P pictures, the inter 8×8 mode, in which four motion vectors are transmitted for an MB, each for an 8×8 subblock. It further specifies the usage of overlapped block motion compensation.

The second and third versions of H.263, which are often called H.263+ and H.263++ [29], respectively, add several optional coding features in the form of annexes. Annex I improves the intra coding by supporting a prediction of intra AC coefficients, defining alternative scan patterns for horizontally and vertically predicted blocks, and adding a specialized quantization and VLC for intra coefficients. Annex J specifies a deblocking filter that is applied inside the motion compensation loop. Annex O adds scalability support, which includes a specification of B pictures roughly similar to those in H.262/MPEG-2 Video. Some limitations of version 1 in terms of quantization are removed by annex T, which also improves the chroma fidelity by specifying a smaller quantization step size for chroma coefficients than for luma coefficients. Annex U introduces the concept of multiple reference pictures. With this feature, motion-compensated prediction is not restricted to use just the last decoded I/P picture (or, for coded B pictures using annex O, the last two I/P pictures) as a reference picture. Instead, multiple decoded reference pictures are inserted into a picture buffer and can be used for inter prediction. For each motion vector, a reference picture index is transmitted, which indicates the employed reference picture for the corresponding block, and it is illustrated in Fig. 11 [36]. The other annexes in H.263+ and H.263++ mainly provide additional functionalities such as the specification of features for improved error resilience.

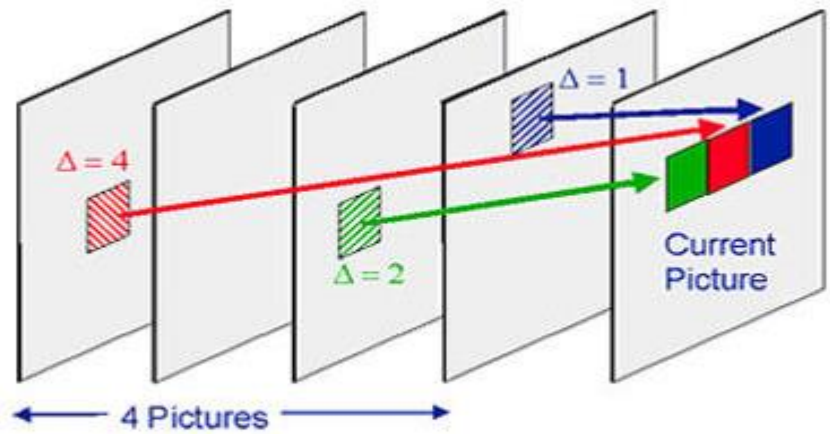


Fig. 11. Multi-frame motion-compensated prediction [36].

The H.263 profiles that provide the best coding efficiency are the Conversational High Compression (CHC) profile and the High Latency profile (HLP). The CHC profile includes most of the optional features (annexes D, F, I, J, T, and U) that provide enhanced coding efficiency for low-delay applications [24].

VII. ITU-T Rec. H.264 | ISO/IEC 14496-10 (MPEG-4 AVC):

H.264/MPEG-4 AVC [14], [15], [32] is the second video coding standard that was jointly developed by ITU-T VCEG and ISO/IEC MPEG. It still uses the concept of 16×16 MBs, but contains many additional features. One of the most obvious differences from older standards is its increased flexibility for inter coding. For the purpose of motion-compensated prediction, an MB can be partitioned into square and rectangular block shapes with sizes ranging from 4 × 4 to 16 × 16 luma samples. H.264/MPEG-4 AVC also supports multiple reference pictures. Similarly to annex U of H.263, motion vectors are associated with a reference picture index for specifying the employed reference picture. The motion vectors are transmitted using quarter-sample precision relative to the luma sampling grid. Luma prediction values at half-sample locations are generated using a 6-tap interpolation filter and prediction values at quarter-sample locations are obtained by averaging two values at integer- and half-sample positions. Weighted prediction can be applied using a scaling and offset for the prediction signal. For the chroma components, a bilinear interpolation is applied. In general, motion vectors are predicted by the component-wise median of the motion vectors of three neighboring previously decoded blocks. For 16×8 and 8×16 blocks, the predictor is given by the motion vector of a single already decoded neighboring block, where the chosen neighboring block depends on the location of the block inside an MB. In contrast to prior coding standards, the concept of B pictures is generalized and the picture coding type is decoupled from the coding order and the usage as a reference picture. Instead of I, P, and B pictures, the standard actually specifies I, P, and B slices. A picture can contain slices of different types and a picture can be used as a reference for inter prediction of subsequent pictures independently of its slice coding types. This generalization allowed the usage of prediction structures such as hierarchical B pictures, shown in Fig. 12 [33], that show improved coding efficiency compared to the IBBP coding typically used for H.262/MPEG-2 Video.

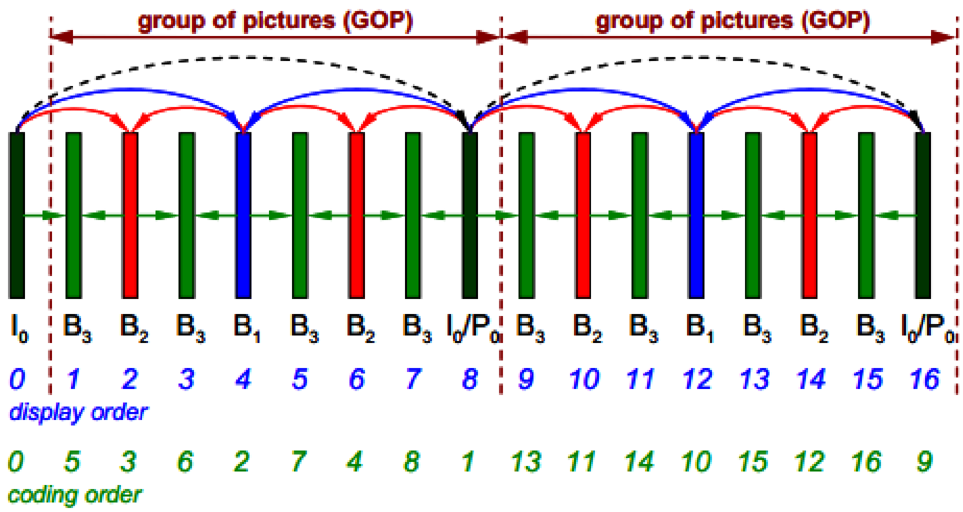


Fig. 12. Hierarchical B picture prediction structure [33].

H.264/MPEG-4 AVC also includes a modified design for intra coding. While in previous standards some of the DCT coefficients can be predicted from neighboring intra blocks, the intra prediction in H.264/MPEG-4 AVC is done in the spatial domain by referring to neighboring samples of previously decoded blocks. The luma signal of an MB can be either predicted as a single 16 × 16 block or it can be partitioned into 4 × 4 or 8 × 8 blocks with each block being predicted separately. Fig. 13 [14] shows intra 4 x 4 luma prediction mode directions.

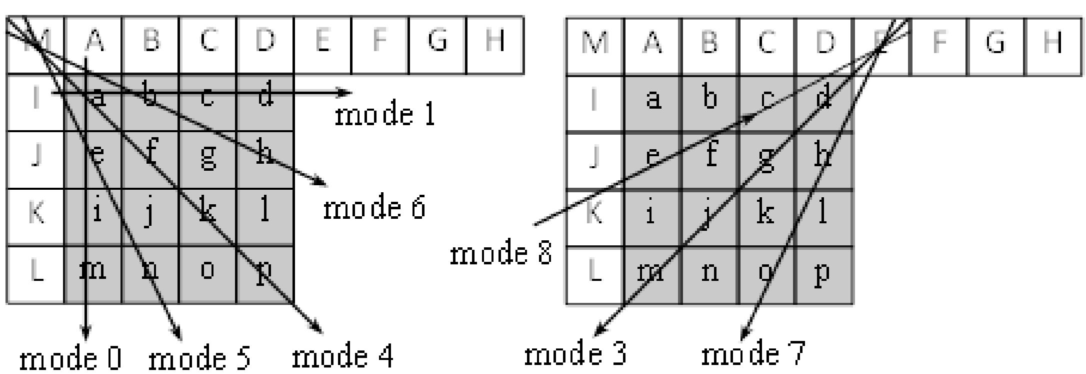


Fig. 13. Intra 4 x 4 luma prediction mode directions (vertical : 0, horizontal : 1, DC : 2, diagonal down left : 3, diagonal down right : 4, vertical right : 5, horizontal down : 6, vertical left : 7, horizontal up : 8) [14].

For prediction of each 8 x 8 luma block, one mode is selected from the 9 modes, similar to the (4x4) intrablock prediction. For prediction of all 16 x 16 luma components of a macroblock, four modes are available. For mode 0 (vertical), mode 1 (horizontal), mode 2 (DC), the predictions are similar with the cases of 4 x 4 luma block. For mode 4 (Plane), a linear plane function is fitted to the upper and left samples.

Each chroma component of a macroblock is predicted from chroma samples above and/or to the left that have previously been encoded and reconstructed. The chroma prediction is defined for three possible block sizes, 8 x 8 chroma in 4:2:0 format, 8 x 16 chroma in 4:2:2 format, and 16 x 16 chroma in 4:4:4 format. The 4 prediction modes for all of these cases are very similar to the 16 x 16 luma prediction modes, except that the order of mode numbers is different: mode 0 (DC), mode 1 (horizontal), mode 2 (vertical), and mode 3 (plane).

For transform coding, H.264/MPEG-4 AVC specifies a 4×4 and an 8×8 transform. While chroma blocks are always coded using the 4 × 4 transform, the transform size for the luma component can be selected on an MB basis. For intra MBs, the transform size is coupled to the employed intra prediction block size. An additional 2×2 Hadamard transform is applied to the four DC coefficients of each chroma component. For the intra 16×16 mode, a similar second-level Hadamard transform is also applied to the 4 × 4 DC coefficients of the luma signal. In contrast to previous standards, the inverse transforms are specified by exact integer operations, so that, in error free environments, the reconstructed pictures in the encoder and decoder are always exactly the same. The transform coefficients are represented using a uniform reconstruction quantizer, that is, without the extra-wide dead-zone that is found in older standards. Similar to H.262/MPEG-2 Video and MPEG-4 Visual, H.264/MPEG-4 AVC also supports the usage of quantization weighting matrices. The transform coefficient levels of a block are generally scanned in a zig–zag fashion, shown in Fig. 14 [14].

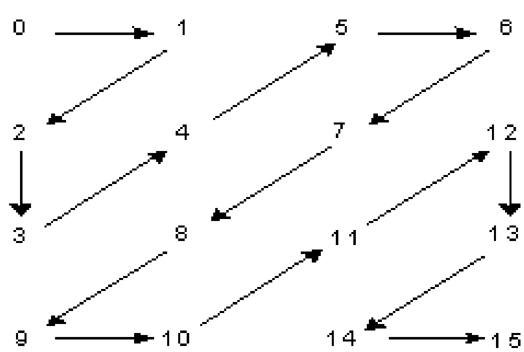


Fig. 14. Zig-zag scan [14]

For entropy coding of all MB syntax elements, H.264/ MPEG-4 AVC specifies two methods. The first entropy coding method, which is known as context-adaptive variable-length coding (CAVLC), uses a single codeword set for all syntax elements except the transform coefficient levels. The approach for coding the transform coefficients basically uses the concept of run-level coding as in prior standards. However, the efficiency is improved by switching between VLC tables depending on the values of previously transmitted syntax elements. The second entropy coding method specifies context-adaptive binary arithmetic coding (CABAC) by which the coding efficiency is improved relative to CAVLC. The statistics of previously coded symbols are used for estimating conditional probabilities for binary symbols, which are transmitted using arithmetic coding. Inter-symbol dependencies are exploited by switching between several estimated probability models based on previously decoded symbols in neighboring blocks. Similar to annex J of H.263, H.264/MPEG-4 AVC includes a deblocking filter inside the motion compensation loop. The strength of the filtering is adaptively controlled by the values of several syntax elements.

The High profile (HP) of H.264/MPEG-4 AVC includes all tools that contribute to the coding efficiency for 8-bit-persample video in 4:2:0 format, and is used for the comparison in this project. Because of its limited benefit for typical video test sequences and the difficulty of optimizing its parameters, the weighted prediction feature is not applied in the testing [24].

VIII. Analysis of Coding Efficiency and Computational Complexity for the HEVC & Other Video Codec:

A. Description of Criteria

The Bjøntegaard measurement method [27] for calculating objective differences between rate-distortion curves was used as evaluation criterion in this section. The average differences in bit rate between two graphs, measured in percent, are reported here. In the original measurement method, separate rate-distortion graphs for the luma and chroma components were used; hence resulting in three different average bit-rate differences, one for each of the components. Separating these measurements is not ideal and is sometimes confusing, as tradeoffs between the performance of the luma and chroma components are not taken into account.

In this method, the rate-distortion graphs of the combined luma and chroma components are used. The combined PSNR ($PSNR_{YUV}$) is first calculated as the weighted sum of the PSNR per picture of the individual components ($PSNR_Y$, $PSNR_U$, and $PSNR_V$), and it is valid for 4:2:0 format only.

$$\mathbf{PSNR_{YUV} = (6 \cdot PSNR_Y + PSNR_U + PSNR_V)/8} \qquad \mathbf{(1)}$$

where $PSNR_Y$, $PSNR_U$, and $PSNR_V$ are each computed as

$$\mathbf{PSNR = 10 \cdot log_{10}((2B - 1)^2/MSE)} \qquad \mathbf{(2)}$$

where B = 8 is the number of bits per sample of the video signal to be coded and the MSE is the sum of squared differences divided by the number of samples in the signal. The PSNR measurements per video sequence are computed by averaging the per-picture measurements.

Using the bit rate and the combined $PSNR_{YUV}$ as the input to the Bjøntegaard measurement method [27] gives a single average difference in bit rate that (at least partially) takes into account the tradeoffs between luma and chroma component fidelities.

B. Results about the Benefit of Some Representative Tools

In general, it is difficult to fairly assess the benefit of a video compression algorithm on a tool-by-tool basis, as the adequate design is reflected by an appropriate combination of tools. For example, introduction of larger block structures has impact on motion vector compression (particularly in the case of homogeneous motion), but should be accompanied by incorporation of larger transform structures as well. Therefore, the subsequent paragraphs are intended to give some idea about the benefits of some representative elements when switched on in the HEVC design, compared to a configuration which would be more similar to H.264/MPEG-4 AVC [24].

In the HEVC specification, there are several syntax elements that allow various tools to be configured or enabled. Among these are parameters that specify the minimum and maximum CB size, TB size, and transform hierarchy depth. There are also flags to turn tools such as temporal motion vector prediction (TMVP), AMP, SAO, and transform skip (TS) on or off. By setting these parameters, the contribution of these tools to the coding performance improvements of HEVC can be gauged.

For the following experiments, the test sequences from classes A to C specified in the Table 5 [37] are used, and frame for each sequence is shown in Fig. 15 [37]. HEVC test model software HM 13.0 [7] is used for these specific experiments. Here in this project, two coding structures are implemented: one suitable for entertainment applications with random access support and one for interactive applications with low-delay constraints.

Class	Resolution in Luma Samples	Sequence	Frame Rate
A	1280 × 720	Kristen And Sara Johnny	60 Hz 60 Hz
B	832 × 480	Race Horses Basketball Drill	30 Hz 50 Hz
C	416 × 240	Blowing Bubbles Basketball Pass	50 Hz 50 Hz

Table 5. Test sequences used in comparison [37]

Kristen And Sara

Johnny

Race Horses

Basketball Drill

Basketball Pass

Blowing Bubbles

Fig. 15. Frame for each sequence [37]

The following tables show the effects of constraining or turning off tools defined in the HEVC MP. In doing so, there will be an increase in bit rate, which is an indication of the benefit that the tool brings. The reported percentage difference in the encoding time is an indication of the amount of processing that is needed by the tool.

Table 6 compares the effects of setting the maximum coding block size for luma to 16×16 or 32×32 samples, versus the 64×64 maximum size allowed in the HEVC MP. These results show that although the encoder spends less time searching and deciding on the CB sizes, there is a significant penalty in coding efficiency when the maximum block size is limited to 32 × 32 or 16 × 16 samples. It can also be seen that the benefit of larger block sizes is more significant for the higher resolution sequences.

	Entertainment Applications		Interactive Applications	
	Maximum CU Size		**Maximum CU Size**	
	32×32	**16×16**	**32×32**	**16×16**
Class A	-	-	7.1%	34.2%
Class B	1.7%	8.0%	2.4%	10.2%
Class C	0.8%	4.1%	1.2%	5.7%
Overall	1.3%	6.1%	3.6%	16.7%
Enc. Time	80%	57%	82%	57%

Table 6. Percentage increment in bit rate for equal PSNR relative to HEVC MP when smaller maximum coding block sizes are used instead of 64 × 64 coding blocks

Table 7 compares the effects of setting the maximum TB size to 8 × 8 and 16 × 16, versus the 32 × 32 maximum size allowed in HEVC MP. The results show the same trend as constraining the maximum coding block sizes. However, the percentage bit-rate penalty is smaller, since constraining the maximum coding block size also indirectly constrains the maximum transform size while the converse is not true. The amount of the reduced penalty shows that there are some benefits from using larger CUs that are not simply due to the larger transforms. It is however noted that constraining the transform size has a more significant effect on the chroma components than the luma component.

	Entertainment Applications		Interactive Applications	
	Maximum Transform Size		**Maximum Transform Size**	
	16×16	**8×8**	**16×16**	**8×8**
Class A	-	-	3.7%	10.3%
Class B	0.8%	3.8%	1.5%	5.5%
Class C	0.3%	2.3%	0.4%	3.0%
Overall	0.6%	3.1%	1.9%	6.3%
Enc. Time	94%	86%	95%	91%

Table 7. Percentage increment in bit rate for equal PSNR relative to HEVC MP when smaller maximum transform block sizes are used instead of 32 × 32 transform blocks

HEVC allows the TB size in a CU to be selected independently of the prediction block size. This is controlled through the RQT, which has a selectable depth. Table 8 compares the effects of setting the maximum transform hierarchy depth to 1 and 2 instead of 3. It shows that some savings in the encoding decision time can be made for a modest penalty in coding efficiency for all classes of test sequences.

	Entertainment Applications		Interactive Applications	
	Max RQT Depth		**Max RQT Depth**	
	2	**1**	**2**	**1**
Class A	-	-	0.3%	0.6%
Class B	0.4%	1.1%	0.3%	1.4%
Class C	0.3%	1.0%	0.3%	1.3%
Overall	0.4%	1.1%	0.3%	1.1%
Enc. Time	90%	81%	92%	83%

Table 8. Percentage increment in bit rate for equal PSNR relative to HEVC MP when smaller maximum RQT depths are used instead of a depth of 3

Table 9 shows the effects of turning off TMVP, SAO, AMP, and TS in the HEVC MP. The resulting bit-rate increase is measured by averaging over all classes of sequences tested. Bit-rate increases of 2.5% and 1.1% were measured when disabling TMVP and SAO, respectively, for the entertainment application scenario. For the interactive application scenario, the disabling of TMVP or SAO tool yielded a bit-rate increase of 2.4%. Neither of these tools has a significant impact on encoding or decoding time. When the AMP tool is disabled, bit-rate increases of 1.0% and 1.3% were measured for the entertainment and interactive applications scenario, respectively. The significant increase in encoding time can be attributed to the additional motion search and decision that is needed for AMP. Disabling the TS tool does not change the coding efficiency.

	Entertainment Applications				Interactive Applications			
	Tools Disabled in MP				Tools Disabled in MP			
	TMVP	**SAO**	**AMP**	**TS**	**TMVP**	**SAO**	**AMP**	**TS**
Class A	-	-	-	-	2.2%	3.2%	1.7%	-0.1%
Class B	2.3%	1.6%	1.0%	0.1%	2.6%	2.8%	1.1%	0.1%
Class C	2.6%	0.5%	0.9%	0.1%	2.3%	1.2%	1.2%	0.0%
Overall	2.5%	1.1%	1.0%	0.1%	2.4%	2.4%	1.3%	0.0%
Enc. Time	98%	99%	86%	96%	100%	100%	87%	96%

Table 9. Percentage increment in bit rate for equal PSNR relative to HEVC MP when the TMVP, SAO, AMP, and TS tools are turned Off

C. Results in Comparison to Previous Standards

For comparing the coding efficiency and computational complexity of HEVC with that of prior video coding standards, coding results for the two different scenarios of entertainment and interactive applications were performed. For HEVC, all coding tools specified in the draft HEVC MP are enabled. For the other video coding standards, H.264/MPEG-4 AVC HP, H.263 CHC for interactive application and H.263 HLP for entertainment application profiles were chosen.

Each test sequence was coded at 5 different bit rates. For HEVC, H.264/MPEG-4 AVC and H.263, the quantization parameter QPI for I pictures was varied in the range from 28 to 44, inclusive.

(1) Interactive Applications:

The first result addresses interactive video applications, such as video conferencing. Since interactive applications require a low coding delay, all pictures were coded in display order, where only the first picture is coded as an I picture and all subsequent pictures are temporally predicted only from reference pictures in the past in display order. The syntax of H.263, H.264/MPEG-4 AVC, and HEVC supports low-delay coding structures that usually provide an improved coding efficiency. Here, class A test sequence (Kristen and Sara) [Table 37] was implemented as an interactive application to find coding efficiency and computational complexity. Fig. 16 and 17 show rate distortion curve and encoding time vs quantization parameter, respectively for interactive applications.

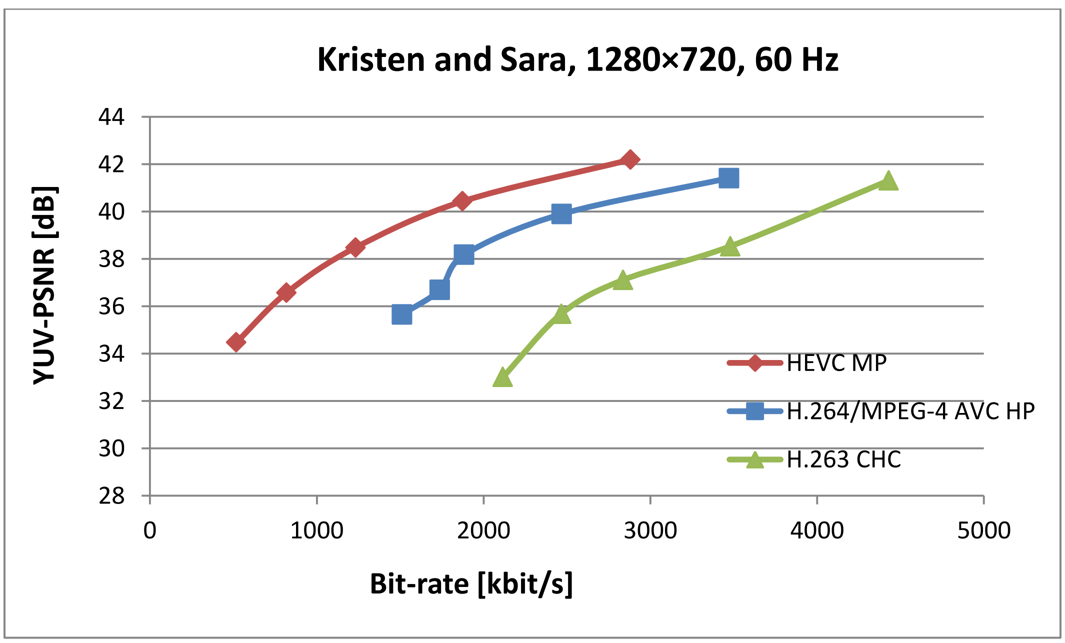


Fig. 16. Rate-distortion curve for interactive applications

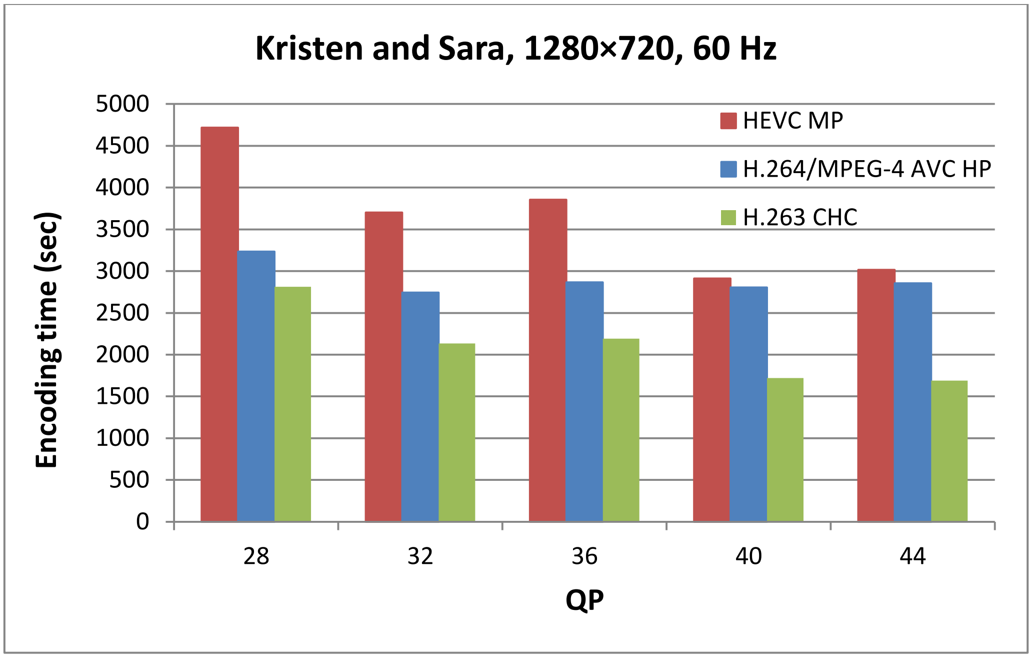


Fig. 17. Encoding time vs quantization parameter for interactive applications

The results from Fig. 16 indicate that the emerging HEVC standard clearly outperforms its predecessors in terms of coding efficiency for interactive application. But, in terms of encoding time, HEVC takes more time to implement; means HEVC is more complex compared to previous video coding standards. The average bit-rate savings between the different codecs, which are computed over the entire test set and the investigated quality range, are summarized in Table 8.

Encoding	Bit-rate savings relative to	
	H.264/MPEG-4 AVC HP	H.263 CHC
HEVC MP	43.86%	59.44%
H.264/MPEG-4 AVC HP	-	27.75%
H.263 CHC	-	-

Table 8. Average bit-rate savings for equal PSNR for interactive applications

(2) Entertainment Applications:

Besides, interactive applications, one of the most promising application areas for HEVC is the coding of high-resolution video with entertainment quality. In contrast to our first experiment, the delay constraints are relaxed for this application scenario. For this application, random access coding structure is used. Here, class B test sequence (Basketball Drill) [Table 37] was implemented as an interactive application to find coding efficiency and computational complexity. Fig. 18 and 19 show rate distortion curve and encoding time vs quantization parameter, respectively for entertainment applications.

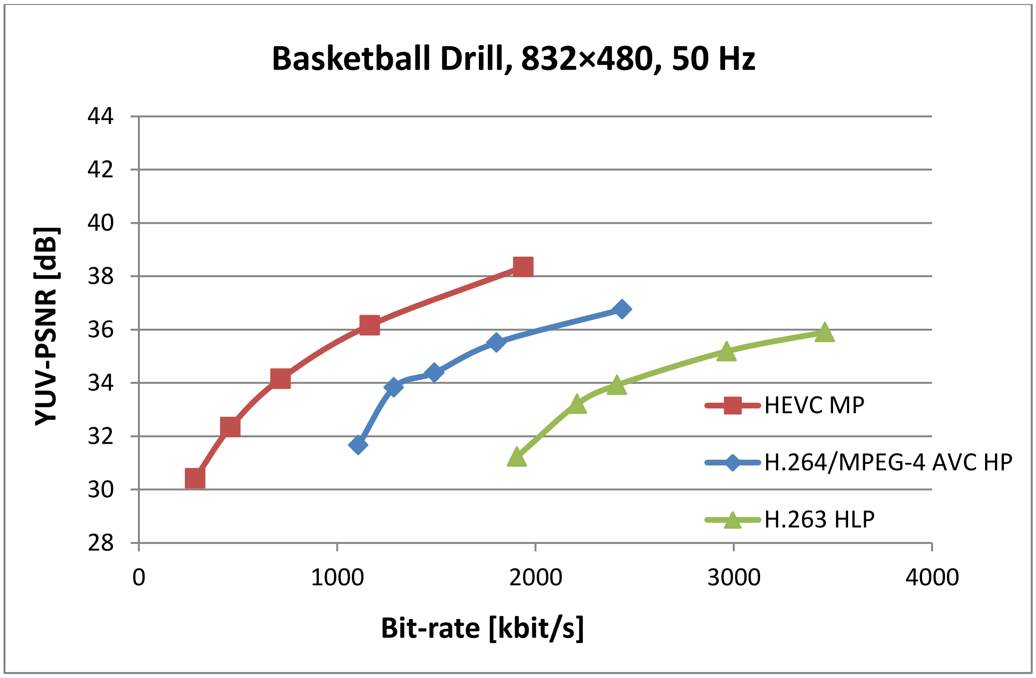


Fig. 18. Rate-distortion curve for entertainment applications

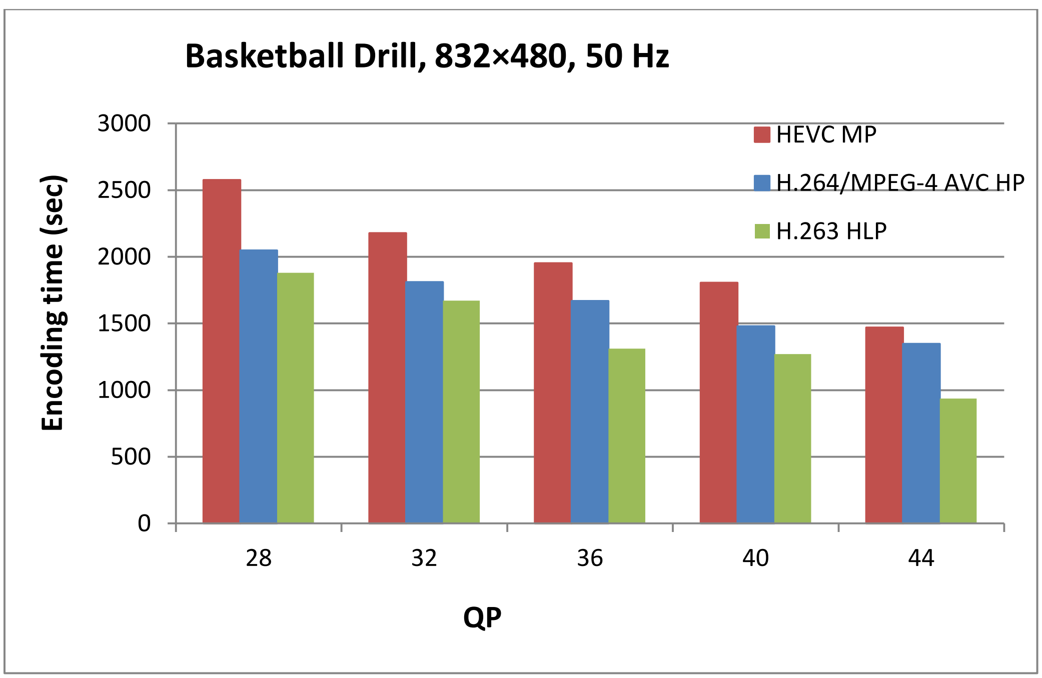


Fig. 19. Encoding time vs quantization parameter for entertainment applications

Encoding	Bit-rate savings relative to	
	H.264/MPEG-4 AVC HP	H.263 HLP
HEVC MP	43.85%	64.79%
H.264/MPEG-4 AVC HP	-	37.28%
H.263 HLP	-	-

Table 9. Average bit-rate savings for equal PSNR for entertainment applications

The bit-rate savings results, averaged over the entire set of test sequences and the examined quality range, are summarized in Table 9. As for the previous case, HEVC provides significant gains in term of coding efficiency relative to the older video coding standards.

IX. Conclusions:

The results documented in this project indicate that the emerging HEVC standard can provide a significant amount of increased coding efficiency compared to previous standards, H.264/MPEG-4 AVC and H.263. The syntax and coding structures of the various tested standards were explained. Special emphasis was given to the various settings and tools of HEVC that are relevant to its coding efficiency. Measurements were then provided for their assessment. PSNR versus bit-rate measurements, encoding time vs quantization parameter and average bit-rate savings for equal PSNR were presented for both interactive and entertainment applications.

X. Future work:

In future, BD-PSNR and BD-bitrate of video coding standards can be compared for both interactive and entertainment applications.

References

[1] F. Pescador et al, “Complexity analysis of an HEVC decoder based on a digital signal processor”, IEEE Trans. on Consumer Electronics, vol. 59, no. 2, pp. 391-399, May 2013.

[2] B. Bross, “High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Consent)”, JCT-VC documents, JCTVC-L1003_v34, Geneva, Switzerland, Jan. 2013.
To access it, go to this link:
http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-L1003_v34 in Number field or type title of this document.

[3] Texas Instruments, OMAP3530 Technical Reference Manual, Literature Number SPRUF98X, June 2012.
To access it, go to this link:
http://www.ti.com/lit/ug/spruf98x/spruf98x.pdf

[4] F. Pescador et al, “An H.264 video decoder based on a DM6437 DSP”, IEEE Trans. on Consumer Electronics. vol. 55, no. 1, pp. 205-212, Feb. 2009.

[5] F. Pescador et al, "A DSP based H.264/SVC decoder for a multimedia terminal," IEEE Trans. on Consumer Electronics, vol. 57, no. 2, pp. 705-712, May 2011.

[6] HEVC Reference Software HM9.0.
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.0rc1/

[7] HEVC Reference Software HM13.0.
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-13.0rc1/

[8] M.T. Pourazad et al, "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC", IEEE Consumer Electronics Magazine, vol. 1, no. 3, pp.36-46, July 2012.

[9] G. J. Sullivan et al, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[10] F. Pescador et al, “On an Implementation of HEVC Video Decoders with DSP Technology”, IEEE International Conference on Consumer Electronics (ICCE), pp. 121-122, Jan. 2013.

[11] G.J. Sullivan et al, “Standardized Extensions of High Efficiency Video Coding (HEVC)”, IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, pp. 1001-1016, Dec. 2013.

[12] F. Pescado et al, “A DSP Based H.264 Decoder for a Multi-Format IP Set-Top Box”, IEEE Trans. on Consumer Electronics, vol. 54, no. 1, pp. 145-153, Feb. 2008.

[13] T. Lindroth et al, “Complexity Analysis of H.264 Decoder for FPGA Design”, IEEE International Conference on Multimedia and Expo”, pp. 1253-1256, July 2006.

[14] S.K. Kwon, A. Tamhankar and K.R. Rao, “Overview of H.264 / MPEG-4 Part 10” J. VCIR, vol. 17, pp. 186-216, April 2006, Special Issue on "Emerging H.264/AVC Video Coding Standard".

[15] K.R. Rao, D. N. Kim and J. J. Hwang, “Video Coding standards: AVS China, H.264/MPEG-4 Part 10, HEVC, VP6, DIRAC and VC-1”, Springer, 2014.

[16] G.J. Sullivan et al, “High efficiency video coding: the next frontier in video compression [Standards in a Nutshell]”, IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 152-158, Jan. 2013.

[17] ITU-T: "H.265 : High efficiency video coding", April 2013.
To access it, go to http://www.itu.int/rec/T-REC-H.265-201304-I/en

[18] H. Lakshman et al, “Generalized Interpolation-Based Fractional Sample Motion Compensation”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 23, no. 3, pp. 455-466, March 2013.

[19] Video lectures from IIT: http://nptel.iitm.ac.in/

[20] F. Pescador et al, “A DSP HEVC decoder implementation based on open HEVC”, IEEE International Conference on Consumer Electronics, pp. 121-122, Jan. 2014.

[21] J. Chen et al, “Planar intra prediction improvement”, JCT-VC document, JCTVC-F483, Torino, Italy, July 2011.
To access it, go to this link:
http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-F483 in Number field or type title of this document.

[22] J. Chen et al, “CE6.a.4: Chroma intra prediction by reconstructed luma samples”, JCT-VC documents, JCTVC-E266, Geneva, Switzerland, March 2011.
To access it, go to this link:
http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-E266 in Number field or type title of this document.

[23] F. Bossen, “HEVC Complexity and Implementation Analysis”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1685-1696, Dec. 2012.

[24] J. R. Ohm et al, “Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1669-1684, Dec. 2012.

[25] J. Vanne et al, “Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1885-1898, Dec. 2012.

[26] G. Corrêa et al, “Performance and Computational Complexity Assessment of High-Efficiency Video Encoders”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1899-1909, Dec. 2012.

[27] G. Bjøntegaard, “Calculation of Average PSNR Differences Between RD Curves”, document VCEG-M33, ITU-T SG 16/Q 6, Austin, TX, Apr. 2001.

[28] H.264/MPEG-4 AVC Reference Software, Joint Model 18.6, Jan. 2014.
Online Available: http://iphome.hhi.de/suehring/tml/download/jm18.6.zip

[29] ITU-T, Video Coding for Low Bitrate Communication, ITU-T Rec. H.263, version 1, 1995, version 2, 1998, version 3, 2000.

[30] ITU-T and ISO/IEC JTC 1, Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 Video), version 1, 1994.

[31] H. Samet, “The quadtree and related hierarchical data structures,” Comput. Survey, vol. 16, no. 2, pp. 187–260, Jun. 1984.

[32] T. Wiegand et al, “Overview of the H.264/AVC video coding standard,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[33] H. Schwarz et al, “Overview of the scalable video coding extension of the H.264/AVC standard,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[34] T. Wiegand et al, “WD2: Working Draft 2 of High-Efficiency Video Coding”, JCT-VC document, JCTVC-D503, Daegu, KR, Jan. 2011.
To access it, go to this link:
http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-D503 in Number field or type title of this document.

[35] G. Côté et al, “H.263+: Video Coding at Low Bit Rates”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 8, no. 7, pp. 849-866, Nov. 1998.

[36] Discussion on “Multi-Frame Motion-Compensated Prediction” by Fraunhofer HHI
To access it, go to this link:
http://www.hhi.fraunhofer.de/en/fields-of-competence/image-processing/research-groups/image-communication/video-coding/multi-frame-motion-compensated-prediction.html

[37] To download test sequences:
1) https://media.xiph.org/video/derf/
2) http://basakoztas.net/hevc-test-sequences/

[38] Special issues on HEVC

1. Special issue on emerging research and standards in next generation video coding, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, pp. 1646-1909, Dec. 2012.

2. Special issue on emerging research and standards in next generation video coding, IEEE Trans. on Circuits and Systems for Video Technology, vol. 23, pp. 2009-2142, Dec. 2013.

3. IEEE Journal of Selected Topics in Signal Processing, vol. 7, pp. 931-1151, Dec. 2013.

[39] H.263 Reference Software: http://www.h263l.com/