An Evaluation of HEVC using Common Conditions

P. Piñol[1], A. Torres[2], O. López[1], M. Martínez[1], and M.P. Malumbres[1]

Abstract— **A new video coding standard, HEVC (High Efficiency Video Coding), has been recently developed. In this paper we will analyze its performance and how it behaves under packet loss conditions, which is a typical scenario for multimedia transmission over wireless networks. We have modified the reference software in order to make it resistant to data loss and we have measured the impact of these losses at different coding settings. For the sake of comparability we have used the Common Conditions which are oftenly used in the developement process to measure the improvements that new contributions produce in the standard.**

Keywords— **HEVC, packet loss, common conditions, error resilience.**

I. Introduction

EARLY this year (25th January 2013), the Joint Collaborative Team on Video Coding (JCT-VC) at their meeting in Geneva agreed on a new video coding standard, informally known as High Efficiency Video Coding (HEVC)[1][2][3]. JCT-VC is formed by members of ISO/IEC Motion Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). This new standard will need half the bit rate than the previous video coding standard H.264/AVC to achieve the same video quality. This increase in compression capability will make High Definition (HD) video feasible for low bandwidth connections. HEVC is also capable to work with video formats that go beyond HD resolution, reaching Ultra High Definition (UHD) video, like 4K UHD (2160p) and 8K UHD (4320p). This new formats and target frame rates, that can reach up to 120 frames per second, bring the need of higher compression efficiency.

In this paper we will evaluate the performance of HEVC for different compression settings. We have modified the reference software [4] to make it resistant to data loss in order to evaluate losses in the bit stream. The original reference software crashes when decoding a bitstream with missing parts. But with our modified decoder version we have been able to test how HEVC bevahes under packet loss conditions and how different settings affect to the reconstructed video quality.

For the sake of comparability we have used the Common Conditions [5] that are frequently used to evaluate the proposed improvements introduced by contributions to the standard. By using this common conditions it is easier to compare the results of video quality and video compression efficiency.

II. HEVC and Common Conditions

In [6] the authors provide a good overview of the new HEVC standard. HEVC is similar in several aspects to H.264/AVC but it also introduces some improvements that lead it to its great performance. It follows the same hybrid compression scheme than H.264/AVC, based on motion or intra estimation/compensation, domain transform followed by coefficient scaling and quantization and, at last, entropy coding.

Some of the novelties in this new video coding standard are the following. In H.264/AVC, the unit used for coding samples is the MacroBlock (MB). A MB consists of 16x16 luma samples and its corresponding chroma samples. In HEVC Coding Tree Units (CTU) may contain Coding Tree Blocks (CTB) of 16x16, 32x32 or 64x64 luma samples and their corresponding chroma samples. Bigger sizes usually lead to improvements in compression, especially in large video formats. Intra prediction (prediction of pixels based on previously coded pixels of the same picture) now supports up to 35 modes instead of the 10 modes of H.264/AVC. This can lead to a better estimation and compensation and can produce smaller residuals which will need less bits to be encoded. A new process is now included in the video coding loop, named Sample Adaptive Offset (SAO). SAO is performed after deblocking. Its aim is to reduce distortion by classifying reconstructed pixels into different categories according to their intensity and edge parameters and then applying a different offset for each category. Also a new structure has been introduced in HEVC: the Video Parameter Set (VPS). VPS permits to send information which is common to the entire video sequence. It works in a similar way to Sequence Parameter Sets (SPS) and Picture Parameter Sets (PPS) which are available in both standards. VPS improves compression and can also be used for error resilience (ER) purposes. Another new feature is the coding using Tiles. Tiles are square regions of a picture that can be decoded indenpendently. This new structure has been introduced in the standard with the goal of facilitating parallel processing. Several other refinements are present in the new standard, like improving the entropy encoder CABAC (Context-Adaptive Binary Arithmetic Coder) and also new features like Wavefront Parallel Processing (WPP). For a deeper look into these and other novelties the reader can check out [6].

In [5] the JCT-VC defines the common test con-

[1]Departamento de Física y Arquitectura de Computadores, Universidad Miguel Hernández de Elche, e-mail: {pablop, otoniel, mmrach, mels}@umh.es
[2]Department of Computer Engineering, Universidad Politécnica de Valencia, e-mail: atcortes@batousay.com

ditions and software reference configurations to be used for HEVC experiments. In that paper it can be found a series of settings in order to evaluate HEVC video codec and to compare the different contributions made to it.

The common test conditions specify a number of video sequences for testing HEVC, grouped in several categories. Five of those categories consist of natural video sequences with different picture resolutions, ranging from 416x260 pixels to 2560x1600 pixels. Another category includes video sequences that have synthetic video in part or in their whole. Frame rates of sequences range from 20 frames per second to 60 frames per second and the bit depth is 8 bits (except for two sequences whose bit depth is 10 bits).

There are four compression modes specified. Each of them is best suited for a class of applications. All Intra (AI) mode codes every frame as an I (intra) frame, this is, no motion estimation/compensation is used. This mode can code a video sequence faster than the other three modes but the compression efficiency that motion compensation can achieve is not exploited here at all. In the other hand, errors in the bitstream are not propagated through the following frames so it provides an inherent mechanism of ER. Applications which are concerned about coding time but not about bandwidth can benefit from this coding mode. Also applications which need to treat each frame independently (like non-linear video editing) are target applications for this mode.

There are two coding modes that hugely increase the compression efficiency and at the same time are concerned about decoding time. They are called Low-Delay P (LP) and Low-Delay B (LB). This two modes use temporal prediction but reference pictures can only be chosen from previous pictures (in display order). In these two modes a coded sequence begins with an I frame and then P frames (for LP mode) or B frames (for LB mode) are inserted until the end of the sequence. P (Prediction) frames can use up to one reference frame and B (Biprediction) frames can use up to two reference frames . Every frame is coded and transmitted in display order so the decoder can display a frame just after receiving and decoding it. This two modes cannot deal with errors in the bitstream because an error in a frame will propagate through succesive frames.

There is a fourth coding mode specified in the common conditions: Random Access (RA) mode. In RA mode an I frame (more specifically, it is a CDR (Clean Decoding Refresh) frame) is inserted every (approximately) one second of video, and the rest are coded as B frames. But here, B frames have reference frames that can appear earlier or later (in display order). So coding (and also decoding) order is not the same as displaying order. This means that coding and decoding includes some delay. Coding delay is caused because the encoder has to wait for future frames (in display order) needed as a reference to encode the present frame. Decoding delay is caused because the decoder does not receive coded frames in display order and has to wait until it has received and decoded all the reference frames needed to decode the next frame. The applications that can use RA mode need to be tolerant to a small delay. On the other hand, inserting an I frame periodically allows actions like fast forwarding/reversing or navigating to a certain moment of the sequence. So this mode is appropriate for applications like video streaming of pre-recorded sequences where navigation is very useful and a little delay can be perfectly asumed.

For each of these four modes, two different bit depths are used for internal calculations: 8 bits (Main) and 10 bits (High). So we have a total of 8 different coding settings combining internal bit depths with coding modes. Configuration files are provided within reference software package [4] [7] [8].

In order to plot Rate-Distortion (RD) curves, four QP (Quantization Parameter) values are specified: 22, 27, 32, and 37. Lower QP values produce bit streams with higher bit rates. At each of this points bit rate and PSNR (Peak Signal-to-Noise) are calculated.

III. Experiments

As it was said before, the original reference software is not resistant to data loss. This means that when the decoder receives an incomplete bit stream it crashes. So, before launching any experiment regarding data losses, we first had to modify the HEVC reference decoder to avoid crashing when some data are missing.

For our experiments we have chosen the sequence named *Race Horses* with a format of 832x480 pixels and a frame rate of 30 frames per second. As dividing a frame in several slices will be used in our future work to introduce ER techniques, we have first measured the overhead produced by varying the number of slices per frame (in the absence of losses). We have performed the rest of our experiments by combining the four different modes (AI, LP, LB, RA) with different number of slices per frame (1, 2, 4, 8, 13, 26) and at the four indicated values for QP (22, 27, 32, 37). Every one of these combinations has been tested at six different packet loss rates (1%, 3%, 5%, 7%, 10%, 20%). For the sake of more realistic randomness, for each one of these loss rates, five different seeds have been used and the values obtained for each PSNR value are the mean values of those five results.

We have first measured the PSNR values obtained for every experiment and then we have implemented a simple error concealment (EC) technique in the decoder to see how it improves video quality in the presence of packet losses and then re-launched all the tests. This simple technique consists basically in filling the missing regions of the current frame with the corresponding regions of the last decoded frame.

After these experiments we also tested two new more modes that we call LP_I32 and LB_I32. These two modes are copies of LP and LB modes but inserting a CDR frame every 32 frames (in the same way as RA does) but keeping reference frames using past ones (in display order) as LP and LB do.

Taking into consideration Bjontegaard work [9][10] we have plotted our RD curves by using *Y-PSNR* and *log(bitrate)*. And also have computed Bjontegaard-Delta (BD) measurements (like BD-PSNR) by using cubic interpolation.

IV. Results

A. Overhead at different slices per frame

In Table I we can see the overhead introduced for every coding mode (at high and low bit rates) when dividing each frame into 2, 4, 8, 13 and 26 slices per frame with respect to using 1 slice per frame. Results are expressed in % of bit rate increase using BD-rate measurement. As it can be seen, overhead for AI mode keeps within certain limits but overhead for RA, LP and LB modes grows rapidly especially for low bit rate settings, because at low bit rates, slice headers represent proportionally a great amount of data. If we plan to divide frames into slices to use ER techniques which add redundancy to the coded video in order to recover information when data loss occurs, then this overhead may not be negligible.

TABLE I
BD-rate increase in % at different slices per frame with High and Low bitrates for each coding mode.

BD-rate	2sl	4sl	8sl	13sl	26sl
AI (High)	0,32	0,85	1,98	2,33	2,87
AI (Low)	0,74	2,05	4,68	5,54	6,97
RA (High)	0,53	1,72	3,80	4,55	6,03
RA (Low)	1,55	4,53	9,96	12,96	19,09
LP (High)	0,27	0,92	2,08	2,60	3,59
LP (Low)	1,04	3,26	7,31	9,83	15,20
LB (High)	0,22	1,00	2,47	3,00	4,12
LB (Low)	1,29	3,47	7,55	10,17	15,45

B. Evaluating HEVC modes without losses

Figure 1 shows the coding efficiency of each of the 4 coding modes specified in common conditions. As it can be seen, AI mode produces much higher bit rates at a same level of quality than RA, LP and LB modes. The most efficient of these three methods is RA. LP has an increase in bit rate of 8.65% over RA, and LB has an increase in bit rate of only 1.95% over RA. RA saves a 62.71% of bit rate with respect to AI. As it was said before, each mode has different target applications which not only depend on coding efficiency.

C. Evaluating HEVC with losses

Figure 2 shows the behavior of LP mode (equivalently LB mode) under data loss conditions. As this mode has only one I frame (the first one), errors propagate continuously through the entire sequence leading to a completely useless video sequence. It does not matter which the loss rate is, even for only 1% of losses, the quality of the reconstructed video is awful.

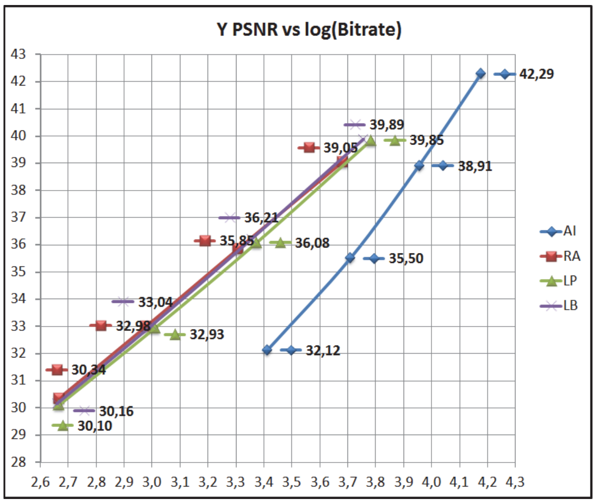


Fig. 1. Compression efficiency of the 4 coding modes.

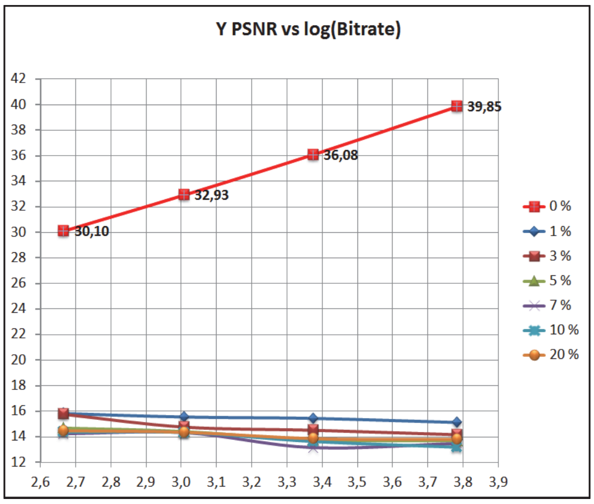


Fig. 2. LP mode at 1 slice per frame and without using EC.

On the opposite situation we find AI mode that inherently has an ER mechanism because no frame uses reference frames, so errors do not propagate at all. In Figure 3 this behavior can be seen for AI mode with 1 slice per frame at different data loss rates. RD curves from 0% to 5% are very close to each other what means that PSNR does not decrease too much.

By observing Figures 3, 4 and 5 a conclusion can be drawn, increasing the number of slices per frame reduces the PSNR value of the reconstructed sequence. Will this conclusion remain valid for sequences reconstructed using our basic EC method?

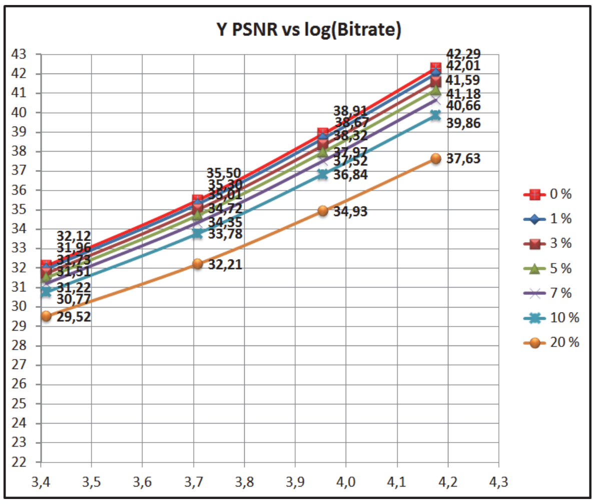


Fig. 3. AI mode at 1 slice per frame and without using EC.

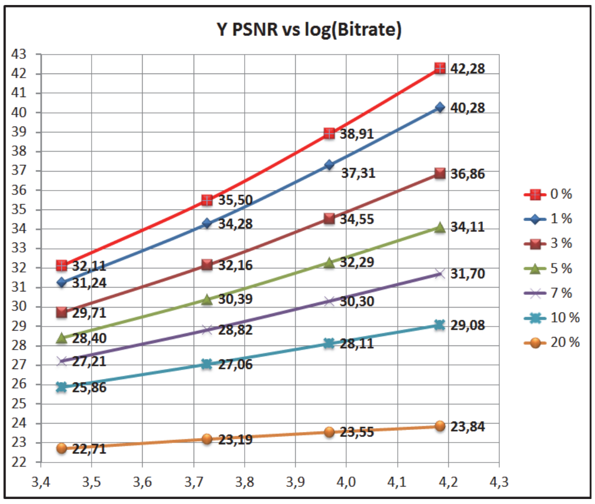


Fig. 5. AI mode at 13 slices per frame and without using EC.

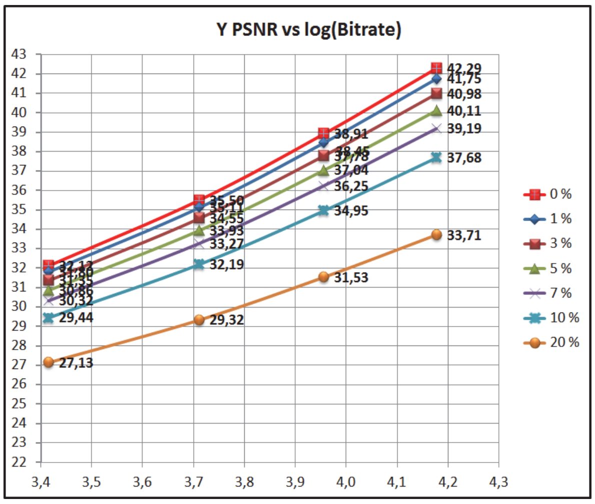


Fig. 4. AI mode at 2 slices per frame and without using EC.

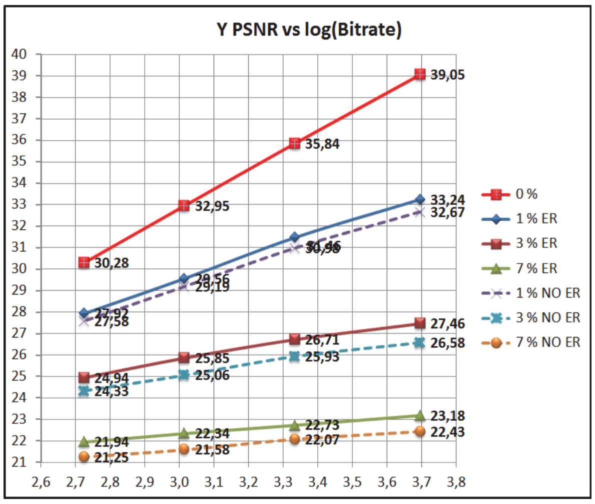


Fig. 6. Comparison of EC and non-EC decoding for RA mode at 13 slices per frame.

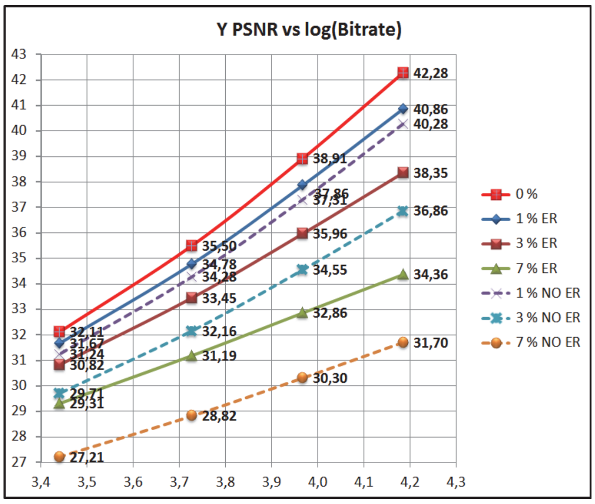
D. *Adding a basic error concealment method*

In Figure 6 we can see the improvement in PSNR that the EC method provides for RA mode at 13 slices per frame at three percentages of data loss. In Figure 7 we can see the corresponding improvements for AI mode. Comparing both figures it can be seen that the simple method implemented is much more effective in AI mode than in RA mode. In RA mode we obtain improvements of 0.44 dB, 0.78 dB and 0.71 dB for 1%, 3% and 7% data loss rates, respectively, while in AI mode we obtain improvements of 0.52 dB, 1.33 dB and 2.42 dB for 1%, 3% and 7% data loss rates, respectively.

In Figure 8 we can find the answer to the previously asked question. This plot represents the RD curves for AI mode, using the basic EC method at 3% data loss rate and varying the number of slices per frame. As in the non concealed version, an increase in the number of slices also produces a worse value of PSNR. The BD-PSNR differences with respect to encoding with 1 slice per frame are the following: -0.30 dB, -0.90 dB, -1,52 dB, -2,10 dB and -2,67 dB for 2, 4, 8, 13 and 26 slices per frame, respectively.

E. *LP_I32 and LB_I32 coding modes*

As we have seen in section IV-C, LB and LP modes do not recover from data losses because they only have one I frame at the beginning. From the second frame till the end of the sequence only P or B frames (which are based in previous frames) will appear. There is not any refreshing frame that can stop error drifting when data loss occurs. This is not true for RA mode, which inserts an I frame (CDR frame) every second of the sequence (for this sequence, every 32 frames). So we have created two new modes (LP_I32 and LB_I32) which are very similar to LP and LB modes but inserting an I frame



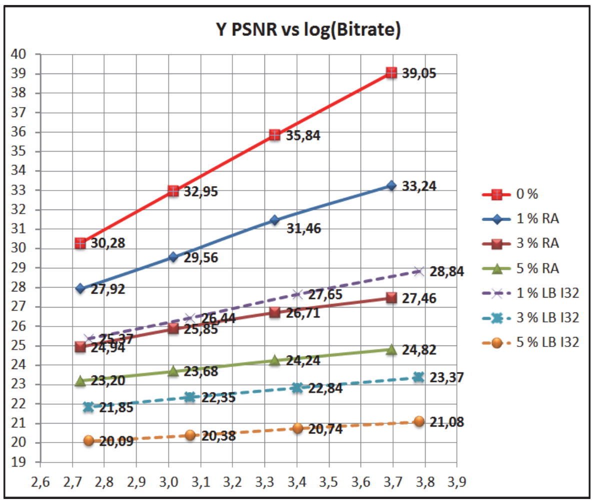


Fig. 7. Comparison of EC and non-EC decoding for AI mode at 13 slices per frame.

Fig. 9. RA mode versus LB_32I mode using EC decoding for different data loss rates.

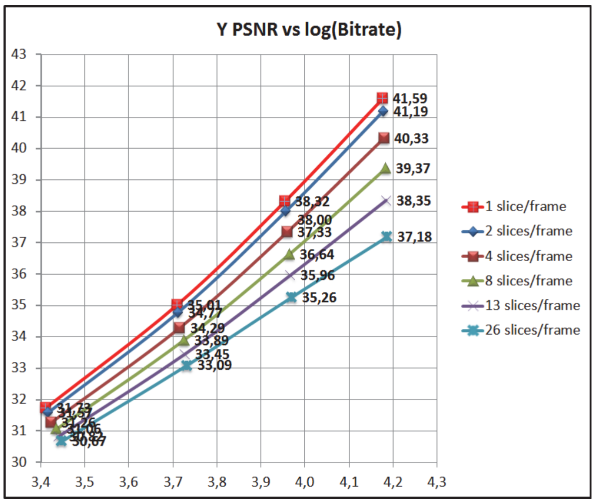


Fig. 8. AI mode at 3% loss using EC decoding for different slices per frame.

(CDR frame) every 32 frames, in a similiar way as RA does. We thought that we would get similar RD curves for LB_I32 mode than for RA mode but in Figure 9 it can be seen that there is a difference in PSNR values for these two modes. By inserting an I frame periodically, PSNR improves over the original LP and LB modes, but we have realised that there is another characteristic of RA mode that also helps to reduce drifting: prediction from future frames (in display order). In RA mode some prediction is done by using future frames, so it can benefit from future I frames to refresh a damaged region. But LB_I32 mode always uses past frames (in display order) so it cannot benefit from future refreshing frames and when an error occurs it will be probably *infect* the following frames. The difference in BD-PSNR for RA and LB_I32 modes is of 3.76 dB, 3.80 dB and 3.49 dB for 1%, 3% and 5% data loss rates.

V. Conclusions

In this paper we have presented an evaluation of HEVC new video coding standard. In order to make results comparable with other experiments we have used the common test conditions, which were designed to measure improvements of the contributions to the standard. As the original reference software is not resistant to data losses we have modified the reference decoder in order it does not crash when some data are missing. By using this modified version we have been able to conduct several tests and see how HEVC and the different coding modes behave under packet losses conditions. We have also implemented a basic error concealment technique in the decoder and shown the benefits of using it. At last, we have modified two of the original coding modes to study what characteristics would help to make a bit stream error resilient.

Acknowledgements

This work was supported by *Spanish Ministerio de Ciencia e Innovación* under project TIN2011-27543-C03-03 and *Consellería de Educación, Formación y Empleo de la Generalitat Valenciana* under project ACOMP/2013/003.

References

[1] High Efficiency Video Coding ITU-T Press Release, http://mpeg.chiariglione.org/sites/default/files/ /files/meetings/docs/w13253_0.doc, ," .

[2] High Efficiency Video Coding ISO/IEC Press Release, http://www.itu.int/net/pressoffice/press_releases/ /2013/01.aspx, ," .

[3] Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, Ye-Kui Wang, and Thomas Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10," Tech. Rep. JCTVC-L1003, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[4] HM Reference Software, https://hevc.hhi.fraunhofer.de/ svn/svn_HEVCSoftware/tags/HM-9.0, ," .

[5] Frank Bossen, "Common test conditions and software reference configurations," Tech. Rep. JCTVC-L1100, Joint

Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[6] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[7] Frank Bossen, David Flynn, and Karsten Suehring, "HM 10.0 Software Manual," Tech. Rep. JCTVC-Software Manual, Joint Collaborative Team on Video Coding (JCT-VC), 2013.

[8] Frank Bossen, David Flynn, and Karsten Suehring, "HEVC HM software development and software technical evaluation (AHG3)," Tech. Rep. JCTVC-M0003, Joint Collaborative Team on Video Coding (JCT-VC), Incheon (Korea), April 2013.

[9] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. VCEG-M33, Video Coding Experts Group (VCEG), Austin (Texas), April 2001.

[10] Gisle Bjontegaard, "Improvements of the BD-PSNR model," Tech. Rep. VCEG-M33, Video Coding Experts Group (VCEG), Berlin (Germany), July 2008.