Wavefront Parallel Processing for AV1 Encoder

Yikai Zhao, Jiangtao Wen
Tsinghua University, Beijing, China
zhaoyk15@mails.tinghua.edu.cn, jtwen@mail.tsinghua.edu.cn

***Abstract*—The emerging AV1 coding standard brings even higher computational complexity than current coding standards, but does not support traditional Wavefront Parallel Processing (WPP) approach due to the lacking of syntax support. In this paper we introduced a novel framework to implement WPP for AV1 encoder that is compatible with current decoder without additional bitstream syntax support, where mode selection is processed in wavefront parallel before entropy encoding and entropy contexts for rate-distortion optimization are predicted. Based on this framework, context prediction algorithms that use same data dependency model as previous works in H.264 and HEVC are implemented. Furthermore, we proposed an optimal context prediction algorithm specifically for AV1. Experimental results showed that our framework with proposed optimal algorithm yields good parallelism and scalability (over 10x speed-up with 16 threads for 4k sequences) with little coding performance loss (less than 0.2% bitrate increasing).**

I. Introduction

AOMedia Video 1 (AV1) is an emerging video coding standard by the Alliance for Open Media (AOM). It's a new, royalty-free video codec format to replace its predecessor VP9 and compete with High Efficiency Video Coding (HEVC) from the Joint Collaborative Team on Video Coding (JCT-VC). Early experiments showed that AV1 encoder can achive same quality as HEVC with 10% to 20% less bitrates with more than 2x increase in computational complexity [1]–[3]. Current reference AV1 Codec Library encodes 720p sequences at less than 0.06 frames per minute in slowest preset. The high encoding complexity makes parallel processing of AV1 encoder more important than ever.

While AV1 standard enables encoder parallelism within a frame between tiles, it lacks parallelism at more fine level of granularity (at superblock level). Tiles partition a picture into rectangular regions for parallel processing. Since Tiles breaks data dependencies for prediction and entropy encoding, it's not optimal for coding performance. Wavefront Parallel Processing (WPP) is another parallelism approach which divide each tile into rows of superblocks for parallel processing. With WPP, the first row is encoded as normal, while all other rows can run encoding at the same time as long as its above row encodes several more superblocks than it. WPP often provides better coding performance than Tiles and can avoid visual artifacts across tile borders [4].

The idea of wavefront parallelization is first introduced by Zhao et al. in [5] for H.264 video encoder. Their analysis on data dependencies showed that for H.264, macroblocks (MBs) from different rows can be processed concurrently as long as its neighbor MBs in its top row have been encoded, which makes it trivial to implement wavefront parallelization in H.264. However, their analysis only applies to encoding configurations using Context-adaptive variable length coding (CAVLC) but not the more powerful Context-adaptive binary arithmetic coding (CABAC), which requires data dependencies in raster order [6]. Though it's optional in H.264, that requirement becomes obligate in newer coding standards like HEVC or AV1 as entropy coding algorithm evolves.

In HEVC, a special syntax element is introduced as standard, indicating whether WPP is enabled, to support CABAC while using WPP [7]. When WPP is enabled in HEVC, each row starts encoding with its own copy of entropy context copied from its above row and decoder can decode in the same way. Many implementations and improvements of WPP in HEVC have been conducted. In [8], Zhang et al. implemented WPP for realtime HEVC encoding on many-core system using ping-pang storage. Chi et al. proposed an improved parallel approach for HEVC based on WPP called Overlapped Wavefront (OWF) [9], Wen et al. took a step further and introduced 3-Dimensional WPP (3D-WPP) in [10]. Those various works showed that WPP and WPP based approaches can achive comparatively high parallelism with little coding performance loss. However, all of them require WPP syntax support in HEVC and cannot be directly adopted by other encoders without WPP syntax support like H.264 or AV1.

In this paper, we propose a novel framework to implement WPP for AV1 encoder in a decoder compatible way, without modification to syntax standard. To achieve that, the mode selection process is seperated from entropy encoding and can be processed parallel using WPP, while the entropy contexts for rate-distortion optimization (RDO) are predicted based on encoded superblocks before actually doing entropy encoding. Two different entropy context prediction algorithms are proposed and implemented to mimic how encoded data are used to calculate entropy contexts for each row in WPP implementations of H.264 and HEVC on previous works respectively. We also propose an optimal context prediction algorithm specifically for AV1. In our experiments, we analyzes coding performance loss, parallelism and scalability of our framework with different context prediction algorithms, as well as parallel approach using Tiles. Experimental results show that our implementation yields better parallelism and scalability, with little impact on coding performance.

The rest of this paper is organized as followings. Data dependencies of AV1 are analyzed in Section II. Our algorithms and implementations of WPP for AV1 are presented in Section III. Section IV contains our experimental results and Section V is the conclusion.

II. Data dependencies in AV1

The use of WPP would limit available encoded data for some superblocks since each row starts encoding before the row above it finishes encoding. Unlike HEVC, AV1 is not designed with WPP in mind so that we need to examine how data dependencie requirement of AV1 affects WPP implementation. In this section, we analyzes data dependencies inside one frame of intra prediction, motion vector prediction and entropy coding. An overview of data dpendencies is illustrated on Fig. 1. As a reference, Fig. 2 illustrates available coded data if WPP is used.

A. Intra Prediction

For each transform block, intra prediction requires reconstructed samples from its top row and left column. Specifically, for a transform block whose top left sample is at (x, y) with size of (w, h), it requires array $CurrentPlane[y-1][x+i]$ and $CurrentPlane[y+i][x-1]$ for $i = 0 \ldots w+h-1$ where $CurrentPlane$ denotes reconstructed pixels of current plane. The required data are further limited by boudary and raster decoding order of current tile, so that for transform blocks at the left-bottom corner of a superblock, pixels from its left-bottom block would not be required.

For a superblock, the maxium possible data dependencies for intra prediction are reconstructed pixels from bottom row of the top superblock and the top-right superblock, as well as right column of the left superblock of the same tile.

B. Motion Vector Prediction

Asides from motion field estimated based on the saved motions from reference frames, motion vectors are also predicted from spartial candidates, which are selected motion vectors of other mode info blocks in the same frame. The locations of candidates are selected for each mode info blocks, they are also limited to not from superblock rows below or superblock columns to the right and must be in the same tile. For a superblock, possible data dependencies for motion vector prediction are mode infos from superblocks at top, left and top-left superblocks of the same tile.

C. Entropy Coding

AV1 uses entropy codec named Boolean Codec to encode entire bitstream except uncompressed headers. A entropy context consists of Cumulative Distribution Functions (CDFs) for each kind of symbol. Multiple entropy contexts are present for each tile or frame, each superblock selects the entropy context to use based on selected modes of its above superblocks on the same column and its left superblocks on the same row, which makes them dependencies of current superblock.

Superblocks of the same tile are encoded in raster order. For each superblock, the prediction mode is selected using RDO based on selected entropy context (CDFs), and the CDFs are in turn updated with symbols of current encoded superblock. In other words, for entropy coding, current encoding superblock depends on all previous superblocks of the same tile in raster order for updating entropy contexts. Each tile of a frame starts with the same entropy contexts and the frame contexts are updated to one of the tile contexts after all tiles finish encoding. There's no data dependencies of entropy coding across tiles.

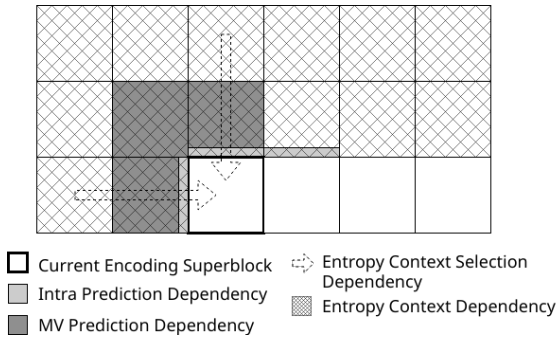


Fig. 1. Data Dependencies of AV1 Encoder

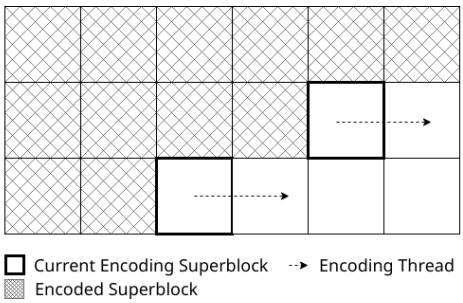


Fig. 2. Data Dependencies Availability of WPP

III. WPP Implementation of AV1

As shown in Fig. 1 and Fig. 2, direct adoption of WPP in AV1 encoder would break data dependencies for entropy encoding. In HEVC standard, when WPP is enabled, a special flag is issued in bitstream and the context model for all rows except the first are inferred from the above row with two superblocks ahead, which removes data dependencies of entire above row. However, that's not possible in AV1 due to the lack of bitstream syntax support. Instead, in this paper we propose a fully standard compatible approach to implement WPP encoding in AV1.

In encoding process, the entropy context is used in two steps for each block: to calculate rate-distortion costs while selecting modes (RDO), and to encode selected mode into bitstream. The entropy context for encoding selected modes to bitstream must match the one used by decoder for correctness so it must be updated in raster order, but the entropy context for calculating rate-distortion costs while selecting modes may not be strictly the same as the one for entropy context. The difference between entropy context for RDO and entropy encoding only affects coding performance (which is the goal of RDO) but not correctness. We propose a encoding framework where the mode selection of each row is processed in parallel

using WPP with best-effort predicted entropy context and the final entropy encoding is then processed sequentially in raster order from the beginning. Formally, we modified the encoding procedure from

procedure AV1 ENCODE
  $C \leftarrow FrameContext$
  for all $Superblock$ in raster order **do**
    $Mode[Superblock] \leftarrow$ Select modes for $Superblock$ based on RDO of C
    Encode $Mode[Superblock]$ into bitstream using C
    $C \leftarrow$ Update CDFs of C using $Mode[Superblock]$
  end for
end procedure

to WPP enabled

procedure AV1 ENCODE WITH WPP
  for all $Superblock$ in raster order **in parallel do**
    $C' \leftarrow$ Predicted context for $Superblock$
    $Mode[Superblock] \leftarrow$ Select mode for $Superblock$ based on RDO of C'
  end for
  $C \leftarrow FrameContext$
  for all $Superblock$ in raster order **do**
    Encode $Mode[Superblock]$ into bitstream using C
    $C \leftarrow$ Update CDFs of C using $Mode[Superblock]$
  end for
end procedure

As a reference, the expected predicted context is the one that is calculated from all previous blocks' mode info in raster order, which is same as the one to encode bitstream (though not possible using WPP), as shown in (1). $CDFs(y, x)$ is the CDFs (entroy context) used while encoding superblock at row y column x of current tile, $\delta(j, i)$ is the symbols from encoded superblock at row j column i of current tile, the $\oplus$ and $\cup$ operator creates CDFs from encoded symbols. w and h are the number of superblocks in one row and one column of current tile respectively.

$$CDFs_{Expected}(y, x) = \bigcup_{j=0}^{y-1} \bigcup_{i=0}^{w-1} \delta(j, i) \oplus \bigcup_{i=0}^{x-1} \delta(y, i) \quad (1)$$

In the following sections, we describe our implementations of context prediction algorithms that mimic previous works in H.264 and HEVC, namely *Copy* Prediction Mode and *Wavefront* Prediction Mode respectively. We also propose our novel algorithm, *Extended* Prediction Mode, and shows that how it is optimal.

A. Copy *Prediction Mode*

Copy Prediction Mode is based on the WPP implementation in H.264 as in [5], it is the most simple and basic one, where each row makes a copy of the frame context and updates its own context as the encoding goes, as denoted by (2).

$$CDFs_{Copy}(y, x) = \bigcup_{i=0}^{x-1} \delta(y, i) \quad (2)$$

Copy Prediction Mode introduces no data dependencies across rows (threads), does not increase any time complexity, but may increase memory usage since N_{thread} copies of entropy contexts needs to be stored.

B. Wavefront *Prediction Mode*

Wavefront Prediction Mode works the same as HEVC for RDO while WPP is enabled. The first row uses the frame context as its context, all other rows make a copy of context from its above row when it starts encoding, which is two blocks behind. In other words, *Wavefront* Prediction Mode is just like *Copy* Prediction Mode except that each row other than the first row copy contexts from its above row instead of frame context, hence they shares the same time complexity and memory usage change. Formally,

$$CDFs_{Wavefront}(y, x) = \bigcup_{j=0}^{y-1} \bigcup_{i=0}^{1} \delta(j, i) \oplus \bigcup_{i=0}^{x-1} \delta(y, i) \quad (3)$$

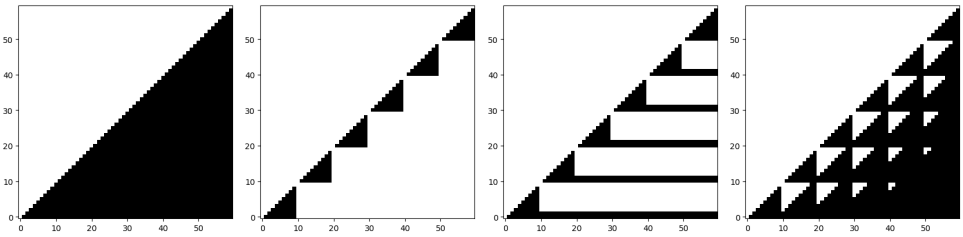
C. Extended *Prediction Mode*

Both *Copy* and *Wavefront* Prediction Mode suffers from the same weakness: none or only a little information of above rows are used to predict entropy context of current row. An notable feature about the entropy context of AV1 is that, the order of most symbols to update the CDFs does not matter (the first 32 symbols have higher weights to make the context adapt faster whose order does matter, other symbols can be unordered). Based on that, we propose *Extended* Prediction Mode specifically for AV1 which overcomes the weaknesses of previous modes by upating multiple contexts after encoding each block. In *Extended* Prediction Mode, each row makes its own copy of entropy context before the encoding starts just like *Copy* Prediction Mode. But for every encoded superblock, the encoding thread updates the context of its own as well as all contexts of its bottom rows using current encoded superblock. For each CDF arrays of a context, a thread mutex is used to protect it from concurrent updating. Given that each row is processed in its own thread and all superblocks has the same encoding time, the effective CDFs used for each superblock is shown as (4).

$$CDFs_{Extended}(y, x) = \bigcup_{j=0}^{y} \bigcup_{i=0}^{\min(x-1+(y-j)\times 2, w-1)} \delta(j, i) \quad (4)$$

Extended Prediction Mode may increases both memory usage and encoding time since multiple contexts needs to be updated (with thread mutex) after each encoded superblock.

Fig. 3 further shows the differences of each context prediction algorithms. The horizonal axis is the index of superblock that is being encoded, every dots of a vertical column shows the indexes of encoded superblocks that is used to calculate CDFs for current encoding superblock. Since a predicted context is used to select modes by RDO which is later encoded using expected context, the more a predicted context differ



(a) Expected (b) *Copy* Mode (c) *Wavefront* Mode (d) *Extended* Mode

Fig. 3. Superblocks used to calculate CDFs for each superblock when $w = 10$, $h = 6$, $N_{thread} = 6$

from the expected one, the much encoding performance would be compromised. Our target for predicted context is to make it use similar coded superblocks to calculate CDFs as expected one for all superblocks.

In conclusion, *Copy* Prediction Mode only use a small subset of expected superblocks to calculate CDFs as it does not use any informations from other rows. *Wavefront* Prediction Mode is slightly better than *Copy* Prediction Mode especially for latter rows since each context is inherented from its above row. *Extended* Prediction Mode shows the optimal result because for each superblock, its CDFs is calculated from all previous encoded superblocks in raster order.

IV. Experimental Results

We have implemented WPP with different described context prediction algorithms based on the reference AV1 Codec Library (version forked on Oct. 2017). Unless otherwise noted, we used constant quality mode and single pass encoding for our experiments (*–end-usage=q -p 1*), other encoding parameters were kept as default. Hardware and software configurations are listed in Table I. Test sequences from the SVT High Definition Multi Format Test Set [11] are used. Note that due to the implemental limitation of the reference library, only horizontal tiles can be concurrently encoded so that experiments of Tiles splits picture into tile columns only.

TABLE I
Experimental setup

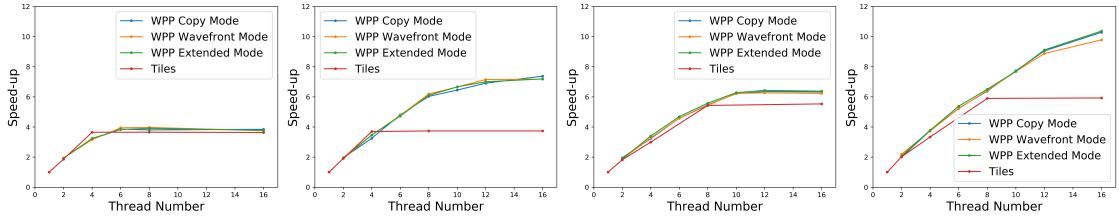
Hardware		Software	
Processor	Intel E5-2683	OS	Ubuntu 16.04.1
Sockets	2	Kernel	4.4.0 generic
Cores/socket	14	Compiler	gcc 5.4.0
Threads/core	2	Compiler Opts	-O3
Frequency (min.)	1200 MHz		
Frequency (max.)	3000 MHz		
L3 Cache	35840K		

To evaluate the coding performance loss of our WPP framework with different context prediction algorithms as well as Tiles of different sizes, we computed the Bjontegaard Delta Bit Rate (BDBR) [12] using average PSNR for luma and chroma components at four different QPs (10, 20, 30, 40) comparing with non-multithreaded, non-tiled coding configuration. Sequences in 720p, 1080p and 2160p resolutions were tested with two possible superblock sizes, 64 and 128. The experimental results are collected in Table II. (Tiles with 8 columns configuration is omitted for 720p sequences due to the limit of minimal size of single tile in AV1.) For Tiles, as we would expect, the coding performance loss increases as number of tiles increases, with more than 1.1% loss for 8 Tile columns. WPP approaches with any context prediction algorithms performs better than Tiles in 4 or more partitions in all test cases. Among WPP implementations with different context prediction algorithms, *Wavefront* Prediction Mode performs better than *Copy* Prediction Mode while *Extended* Prediction Mode being the best in all test cases, with up to 1.2% bitrate saving over *Copy* Prediction Mode at the same quality. WPP approach with our AV1 specifical optimal *Extended* Prediction Mode achives around only 0.2% loss for all test cases, which is better than Tiles with any number of partitions.

Fig. 4 illustrates the speed-up of WPP with different prediction algorithms in different number of threads, as well as Tiles in different number of tile columns with the same number of threads as columns. Two different resolutions (1080p and 2160p) with two different allowed superblock size (64 and 128) are tested. For all test cases, result shows that WPP has better parallelism than tiles, especially as number of threads increases since the number of tiles is limited by the minium size of a tile, which sets a upper boundary of its scalability. For sequences in 2160p with superblock size of 64, WPP approaches have almost linear scalability, with 9x speed-up with 12 threads, 10.5x speed-up with 16 threads. As a reference, we share similar parallelism results with WPP in HEVC which can achieve average speedups of 8.7 in 12-core system for 2160p sequences [9]. The scalability of WPP is affected by both picture resolution and size of superblocks, smaller height of frame and bigger superblocks leads to less available rows of superblocks for parallel processing. Also, since WPP yields low parallelism at the start and end of each tile due to its "wavefront" nature, smaller width of frame further affects the scalability. However, as parallel processing is usually used for high definition sequences, the impact of these cases should be minor. WPPs with different contexts prediction algorithms shares similar results, our proposed optimal *Extended* Prediction Mode does not introduces noticable extra overhead.

TABLE II
CODING PERFORMANCE LOSS COMPARISION, BDBR (%)

SB Size	Sequence Name	Sequence Format	Sequence Frames	WPP Only *Copy*	WPP Only *Wavefront*	WPP Only *Extended*	Tiles Only 2 Cols	Tiles Only 4 Cols	Tiles Only 8 Cols
128	ParkJoy	720p50	300	1.327	0.462	**0.146**	0.511	1.467	-
128	DucksTakeOff	1080p50	300	1.094	0.535	**0.192**	0.498	1.319	-
64	DucksTakeOff	1080p50	300	1.297	0.990	**0.258**	0.450	1.149	-
128	CrowdRun	2160p50	100	0.607	0.357	**0.183**	0.290	0.657	1.238
64	CrowdRun	2160p50	100	0.658	0.601	**0.211**	0.331	0.669	1.150



(a) DucksTakeOff 1080p, sb=128 (b) DucksTakeOff 1080p, sb=64 (c) CrowdRun 2160p, sb=128 (d) CrowdRun 2160p, sb=64

Fig. 4. Parallelism and scalability comparision

V. CONCLUSION

In this paper, we presented a novel WPP framework for AV1 encoder. Though WPP has been implemented in H.264 and HEVC by previous works, they are not compatible with AV1: H.264 with CAVLC does not require data dependencies in raster order while AV1's entropy encoder does; HEVC has a special flag in bitstream syntax to make decoder use different entropy contexts for each row when WPP is enabled, which AV1 lacks. Our proposed framework seperated mode selection process from entropy encoding so that mode selection can be processed in wavefront parallel while entropy context for RDO is predicted.

Based on the framework, different context prediction algotithms are implemented and analyzed. *Copy* Prediction Mode and *Wavefront* Prediction Mode are two algotithms that ultilize the same data dependency model as previous works in H.264 and HEVC respectively. Experiments shows that these two algorithms introduce great coding performance loss (up to 1.3%) which is sometimes worse than Tiles approach, since little coded information across rows are exploited.

We then proposed a new optimal context prediction algorithms on our WPP framework for AV1 specifically, *Extended* Prediction Mode. It leverages all previous encoded superblocks and is theoretical optimal. Experiments showed that this algorithm performs better than all previous approaches as well as Tiles at different partition size, with around 0.2% coding performance loss comparing to non-multithreaded non-tiled one. It also has significant better parallelism and scalability for high definition sequences than Tiles (over 10x speed-up with 16 threads for 4k sequences). We conclude that our proposed WPP framework with optimal context prediction algorithm brings AV1 a better parallel processing approach than existing Tiles in terms of both performance and scalability in a syntax compatible way.

REFERENCES

[1] D. Grois, T. Nguyen, and D. Marpe, "Coding efficiency comparison of av1/vp9, h. 265/mpeg-hevc, and h. 264/mpeg-avc encoders," in *Picture Coding Symposium (PCS), 2016*. IEEE, 2016, pp. 1–5.

[2] J. Ozer, "A102: How to: Comparing and choosing the best hevc codec," in *Streaming Media East 2017*, 2017.

[3] K. P. Sriram Sethuraman, Cherma Rajan A., "Analysis of the emerging aomedia av1 video coding format for ott use-cases," *International Broadcasting Convention*, 2017.

[4] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[5] Z. Zhao and P. Liang, "Data partition for wavefront parallelization of h. 264 video encoder," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 2006, pp. 4–pp.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[7] R. H. ITU-T, "265 (04/13)," *Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services–Coding of Moving Video, High Efficiency Video Coding, Online: http://www. itu. int/rec/T-REC-H*, pp. 265–201 304.

[8] S. Zhang, X. Zhang, and Z. Gao, "Implementation and improvement of wavefront parallel processing for hevc encoding on many-core platform," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[9] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel scalability and efficiency of hevc parallelization approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1827–1838, 2012.

[10] Z. Wen, B. Quo, J. Liu, J. Li, Y. Lu, and J. Wen, "Novel 3d-wpp algorithms for parallel hevc encoding," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1471–1475.

[11] L. Haglund, "The svt high definition multi format test set," *Swedish Television Stockholm*, 2006.

[12] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," in *ITU-T Q. 6/SG16 VCEG, 15th Meeting, Austin, Texas, USA, April, 2001*, 2001.