Applied Mathematical Sciences, Vol. 7, 2013, no. 137, 6803 - 6824
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2013.311644

Intra Compression Efficiency in VP9 and HEVC

Maxim P. Sharabayko

Postgraduate at Tomsk Polytechnic University
Junior Research Fellow at Tomsk State University of Control Systems and Radioelectronics
634050 Tomsk, Russia
maxim.sharabayko@elecard.ru

Oleg G. Ponomarev

Assistant professor at Tomsk State University
Senior Research Fellow at Tomsk State University of Control Systems and Radioelectronics
634050 Tomsk, Russia
oleg.ponomarev@elecard.ru

Roman I. Chernyak

Postgraduate at Tomsk State University
Junior Research Fellow at Tomsk State University of Control Systems and Radioelectronics
634050 Tomsk, Russia
roman.chernyak@elecard.ru

Abstract

The amount of video data stored on local devices or transmitted over the networks is permanently increased. The emerging of a more efficient next generation video coding standard is of a high demand at the moment. There seem to be two main contenders for the position of the next state-of-the-art video compression standard: JCT-VC H.265/HEVC and Google VP9. The announced aim of HEVC is to achieve twice more efficient compression compared to H.264/AVC, and VP9 was developed to get half the bit-rate of VP8 with royalty-free video

codec. Intra compression is one of the main features that determines the compression efficiency of the whole codec.

In this paper we get into detailed overview of intra compression data-flow in HEVC and VP9. We describe common and unique stages of both standards. Then we carry out experiments with JCT-VC HM and WebM VP9 encoders on intra compression efficiency. We also turn some of the HEVC features off to get its dataflow as close to VP9 as possible. Finally we get into discussion of the efficiency of both codecs, the corresponding standards and their intra compression algorithms.

Mathematics Subject Classification: 94A08

Keywords: video compression, intra-prediction, Planar prediction, True Motion, H.265/HEVC, VP9

1 Introduction

The current industrial video compression standard H.264/AVC was adopted in 2003. It provides a superior video compression efficiency compared to other existing and widely spread standards such as MPEG2 or VP8. However the amount of video data stored on local devices or transmitted over the networks is permanently increased. According to Cisco [2] mobile video traffic was 51 percent of the entire global Internet traffic by the end of 2012 and it is expected to be 66 percent by 2017. The ability to get better compression rates will eventually decrease network bandwidth load.

The emerging of a more efficient next generation video coding standard is of a high demand at the moment. There seem to be two main contenders for the position of the next state-of-the-art video compression standard: JCT-VC H.265/HEVC and Google VP9. HEVC is being developed by JCT-VC group - the creators of AVC. It is an evolution of AVC concepts with some innovations. On the other hand, VP9 is a Google initiative to get a royalty-free compression standard with efficiency superior to AVC. It expands techniques used in AVC and VP8 and is very likely to replace AVC at least in the YouTube video service.

The basis of any video compression standard is intra-frame coding that determines the resulting compression efficiency. In this paper we present a detailed overview of intra-frame compression techniques used in VP9 and HEVC. Then we analyze their unique parts and carry out experiments on intra-frame compression efficiency of HM and WebM VP9 encoders.

2 General compression dataflow

Both HEVC and VP9 video compression standards are hybrid block-based codecs relying on spatial transformations. General compression dataflow of hybrid block-based encoders is illustrated in Fig.1. The input video frame is initially partitioned into blocks of the same size called macroblocks. The compression and decoding process works within each macroblock. A macroblock is subpartitioned into smaller blocks to perform prediction. There are two basic types of prediction: intra and inter. Intra-prediction works within a current video frame and is based upon the compressed and decoded data available for the block being predicted. Inter-prediction is used for motion compensation: a similar region on previously coded frames close to the current block is used for prediction. The aim of the prediction process is to reduce data redundancy and, therefore, not store excessive information in coded bitstream.

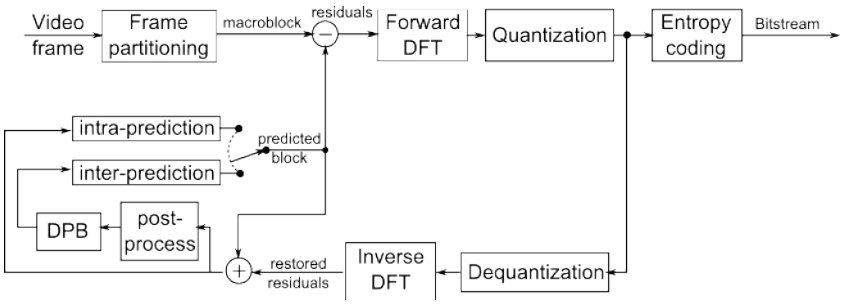


Figure 1: Hybrid block-based codec dataflow

Once the prediction is done, it is subtracted from the original data to get residuals that should be compressed. Residuals are subject to Forward Discrete-Fourier Transform (DFT). DFT translates spatial residual information into frequency domain. Thus the remaining spatial redundancy of this information is partly reduced. Quantization is applied to the transformed matrix to lose insufficient information. The insufficiency threshold is predetermined by encoder configuration. The remaining data and the steps applied are subject to entropy coding, which makes it possible to get compressed bitstream.

For inter- and intra-prediction purposes the compressed data should be restored in the encoder. The only data loss takes place after integer DFT and quantization. Dequantization and inverse DFT are performed to restore residuals. Then the restored residuals and the predicted values are summed up to get restored pixel values, identical to those achieved in the decoder. These restored values are used for intra-prediction within current video frame. An additional frame post-processing stage is optionally applied to eliminate image blockiness introduced by DFT and quantization. The final restored and

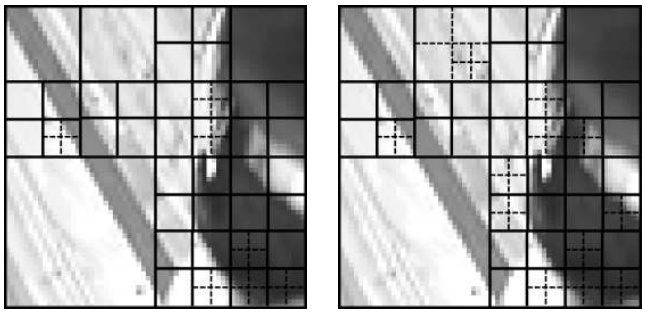
post-processed video frame is stored in Decoded Picture Buffer (DPB) for inter prediction of further frames.

VP9 and HEVC both utilize the described general compression dataflow, but differ in details.

3 Intra compression in HEVC

3.1 Macroblock concept

The concept of macroblock in HEVC [5] is represented by the Coding Tree Unit (CTU). CTU size can be 16×16, 32×32 or 64×64, while AVC macroblock size is 16×16. Larger CTU size aims to improve the efficiency of block partitioning on high resolution video sequences (bitrate savings are about 16% [6]). Larger blocks provoke the introduction of quad-tree partitioning (Fig. 2) of a CTU into smaller coding units (CUs). A coding unit is a bottom-level quad-tree syntax element of CTU splitting. The CU contains a prediction unit (PU) and a transform unit (TU).



a) *b)*

Figure 2: CTU splitting example with solid lines for CU split: a) with PU splitting depicted as dotted lines; b) with TU splitting depicted as dotted lines

The TU is a syntax element responsible for storing transform data. The CU can be split in TUs in a quad-tree structure down to the smallest TU size available. Allowed TU sizes are 32×32, 16×16, 8×8 and 4×4 with the respectively sized DFT matrix.

The PU is a syntax element to store prediction data like the intra-prediction angle or inter-prediction motion vector. The CU can contain up to four prediction units. CU splitting on PUs can be 2N×2N, 2N×N, N×2N, N×N, 2N×nU, 2N×nD, nL×2N and nR×2N (Fig. 3) where 2N is a size of a CU being split. In the intra-prediction mode only 2N×2N PU splitting is allowed. An N×N PU split is also possible for a bottom level CU that cannot be further split into sub CUs.

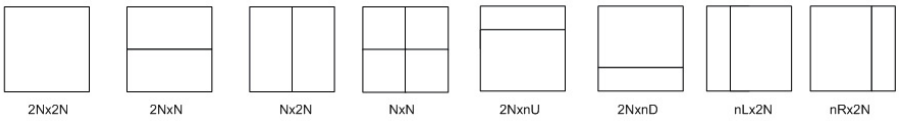


Figure 3: PU splitting

3.2 Intra-prediction modes

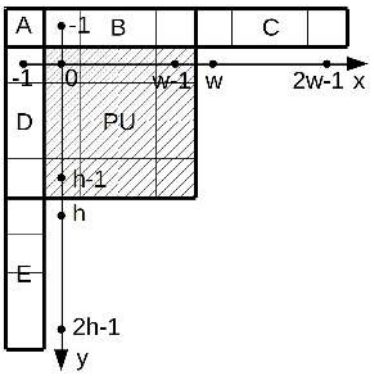


Figure 4: Prediction unit

To describe intra-prediction modes in HEVC let us assume the block being predicted is a pixel matrix $P = \{p(x, y)\}$, where $x = 0, \ldots, w - 1$ and $y = 0, \ldots, h - 1$, with the size $w \times h$ (Fig. 4). Intra-prediction in HEVC is always performed on a square-sized pixel matrix, therefore let $w = h = s$. intra-prediction of PU pixels may involve below-left (set $E = \{p(-1, y)\}$ when $y = s, \ldots, 2 \cdot s - 1$), left (set $D = \{p(-1, y)\}$ when $y = 1, \ldots, s - 1$) above-left (set $A = \{p(-1, -1)\}$), above (set $B = \{p(x, -1)\}$ when $x = 1, \ldots, s - 1$) and above-right (set $C = \{p(x, -1)\}$, $x = s, \ldots, 2 \cdot s - 1$) neighboring pixels. The availability of those pixels is determined by PU positioning.

To perform HEVC intra-prediction first the intra-prediction pattern $R = \{r(i)\}$ with $i = -2 \cdot s, \ldots, 2 \cdot s$ should be formed in the following way:

$$r(i) = \begin{cases} p(-1, -1 - i) & if\ i < 0 \\ p(i - 1, -1) & if\ i \geqslant 0 \end{cases}$$

In some cases (see Table 1) for luma blocks a filtered intra-prediction pattern $R' = \{r'(i)\}$ with $i = -2 \cdot s, \ldots, 2 \cdot s$ is used, where

$$r'(i) = \begin{cases} \frac{1}{2} \cdot (r(i) + r(i + 1)) & if\ i = -2 \cdot s \\ \frac{1}{2} \cdot (r(i - 1) + r(i)) & if\ i = 2 \cdot s \\ \frac{1}{4} \cdot (r(i - 1) + 2 \cdot r(i) + r(i + 1)) & otherwise \end{cases}$$

In the following description of the intra-prediction process we address the intra-prediction pattern just as $r(i)$ considering that $r'(i)$ should be used in some cases, determined in Table 1.

Table 1: Intra-prediction pattern filtering cases

Mode	planar	DC	2-8	9	10	11	12-17	18	19-24	25	26	27	28-34
4×4	yes	no	no	no	no	no	no	no	no	no	no	no	no
8×8	yes	no	no	no	no	no	no	yes	no	no	no	no	no
16×16	yes	no	yes	no	no	no	yes	yes	yes	no	no	no	yes
32×32	yes	no	yes	yes	no	yes	yes	yes	yes	yes	no	yes	yes

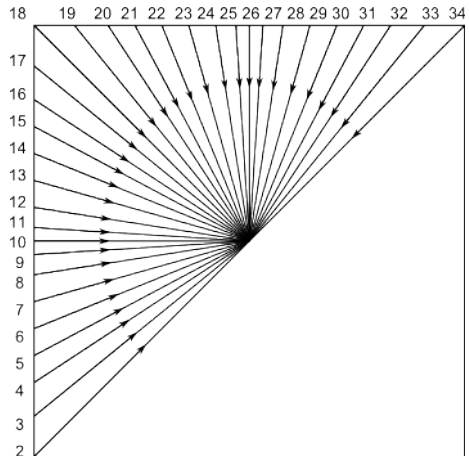


Figure 5: HEVC angular intra-prediction modes

There is a total of 35 intra-prediction modes in HEVC: planar (mode 0), DC (mode 1) and 33 angular modes (modes 2-34 on Fig.5).

DC intra-prediction is the most simple mode in HEVC. All PU pixels are set equal to the mean value of all available neighboring pixels from sets B and D. Thus the predicted pixel value is:

$$p(x,y) = DC = \frac{1}{2 \cdot s} \cdot \left(\sum_{i=-s}^{-1} r(i) + \sum_{i=1}^{s} r(i) \right)$$

The left column $p(0,y)$ and the top row $p(x,0)$ pixel values of PU are subjected to additional linear filtering:

$$p(0,y) = \frac{1}{4}(r(-y-1) + 3 \cdot DC + 2)$$

and

$$p(x,0) = \frac{1}{4}(r(x+1) + 3 \cdot DC + 2).$$

This operation is performed only for PU sizes 4×4, 8×8 or 16×16 and aims to smooth the transition from neighboring pixels to the DC value.

Planar intra-prediction is the most computationally expensive. It is a two-dimensional linear interpolation. Fig.6 illustrates an example of planar prediction process. Each predicted pixel value $p(x, y)$ is determined like:

$$p(x, y) = \frac{1}{2 \cdot s} \left(h(x, y) + v(x, y) + s \right)$$

where

$$h(x, y) = (s - x - 1) \cdot r'(-y - 1) + (x + 1) \cdot r'(s)$$

is a horizontal interpolation of pixel value and

$$v(x, y) = (s - y - 1) \cdot r'(x + 1) + (y + 1) \cdot r'(-s)$$

is a vertical interpolation of pixel value.

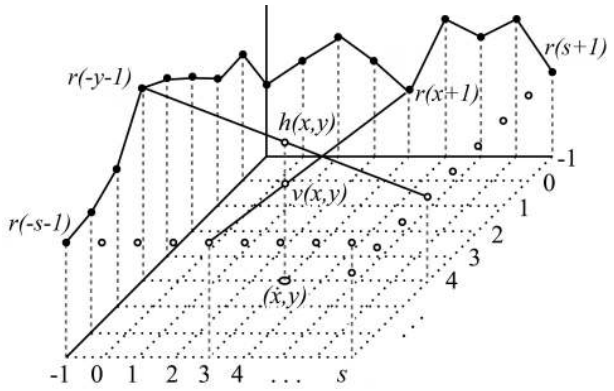


Figure 6: Illustration of planar intra-prediction interpolation process

Angular intra-prediction modes 2-34 are linear interpolations of pixel values in the corresponding directions (Fig. 5). Vertical intra-prediction (modes 18-34) is an up-down interpolation of neighboring pixel values. Let φ be an angle between the vertical y-axis and the interpolation direction (Fig. 7). Therefore it would have positive values for clockwise directions and negative values for counterclockwise directions. If the (x, y) position is between two reference pixels, then the predicted $p(x, y)$ value is linearly interpolated with a $1/32$ pixel accuracy. The vertically predicted pixel value is determined like:

$$p(x, y) = \left\lfloor \frac{(32 - \Delta) \cdot r''(x + 1 + i) + \Delta \cdot r''(x + 2 + i) + 16}{32} \right\rfloor,$$

for $x = 0, \ldots, s - 1$ and $y = 0, \ldots, s - 1$, where $i = \frac{32 \cdot (y+1) \cdot tg(\varphi)}{32}$, $\Delta = \lfloor 32 \cdot (y + 1) \cdot tg(\varphi) \rfloor - 32 \cdot i$, and $\lfloor \cdot \rfloor$ is an integer floor of a value. The reference

sample $r''(i)$ is formed the following way:

$$r''(x) = \begin{cases} r(x+1) & x \geqslant -1 \\ r(-\left\lfloor \frac{1}{256} \cdot (256 \cdot x \cdot ctg(\varphi)) \right\rfloor - 1) & x < -1 \; and \; \varphi < 0 \end{cases}$$

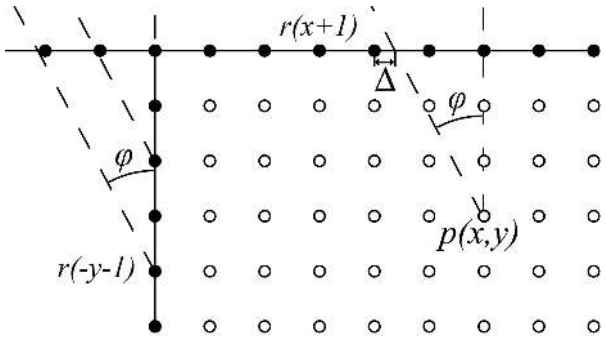


Figure 7: Vertical intra-prediction interpolation

Horizontal intra-prediction (modes 2-17) is basically the same as vertical, except the prediction direction is left-to-right and angle φ is an angle between the horizontal x-axis and the interpolation direction, taking positive values in counterclockwise directions. The predicted pixel value $p(x, y)$ is determined by:

$$p(x,y) = \left\lfloor \frac{(32 - \Delta) \cdot r''(y+1+i) + \Delta \cdot r''(y+2+i) + 16}{32} \right\rfloor,$$

for $x = 0, \ldots, s-1$ and $y = 0, \ldots, s-1$, where $i = \frac{32 \cdot (x+1) \cdot tg(\varphi)}{32}$, $\Delta = \lfloor 32 \cdot (x+1) \cdot tg(\varphi) \rfloor - 32 \cdot i$. The reference sample $r''(i)$ is formed the following way:

$$r''(x) = \begin{cases} r(-x-1) & y \geqslant -1 \\ r(\left\lfloor \frac{1}{256} \cdot (256 \cdot y \cdot ctg(\varphi)) \right\rfloor + 1) & y < -1 \; and \; \varphi < 0 \end{cases}$$

It is worth mentioning, that intra-prediction modes 2, 10, 18, 26 and 34 have $\Delta = 0$ and the prediction takes only one reference pixel for one predicted pixel. Also prediction modes 10 and 26 do not use a filtered intra-prediction pattern, but perform additional border filtering in cases of luma prediction of sizes 4×4, 8×8 or 16×16. For horizontal mode 10 the top row of predicted pixels is:

$$p(x,0) = r(-1) + \frac{1}{2} \cdot (r(x+1) - r(0))$$

and for vertical mode 26:

$$p(0,y) = r(1) + \frac{1}{2} \cdot (r(-y-1) - r(0))$$

3.3 Transform and quantization

The residuals after subtraction of predicted pixel values are subjected to DFT and quantization. HEVC has an integer approximation of discrete cosine transform and discrete sine transform (DCT II and DST VI by classification in [1] respectively). A transform operation is specified for the 32×32, 16×16, 8×8 and 4×4 pixel matrix. DST is used for 4×4 intra coded blocks. DCT transform matrices utilize symmetry properties that allow for using parts of the "butterfly" structures used for fast FFT-like algorithms to reduce the number of multiply/adds [3].

The quantization is inherited from AVC. The modifications are due to the introduction of additional transform sizes.

3.4 Entropy coding

Entropy coding in HEVC is an evolution of context-adaptive binary arithmetic coding (CABAC) in AVC. The HEVC arithmetic coder is called syntax-based context-adaptive binary arithmetic coder (SBAC) and has 27 context models for syntax elements coding. Each context is adaptively changed based on the data already processed. The principal modifications in SBAC compared to CABAC are related to the provision of parallel coding and decoding possibility.

3.5 Post-processing

HEVC has two adaptive post-processing stages: deblock filtering (DBF) and Sample-Adaptive Offset (SAO). Both stages are optional and can be turned off.

The aim of the DBF is to reduce blockiness artifacts introduced after quantization of transformed coefficients. The filtering is applied at TU boundaries with the absolute position multiple to 8. The vertical boundaries are processed first, and the horizontal boundaries - second. A filtering algorithm is adaptive to fluctuations of pixel values near the boundaries. If the fluctuation is low, then the "strong" filter is applied affecting 3 pixels on both sides of the boundary. If the fluctuations are high, then the "weak" filtering is applied affecting 2 pixels from both sides. When the boundary pixel value variance within boundary segment is higher than the threshold, no filtering is applied.

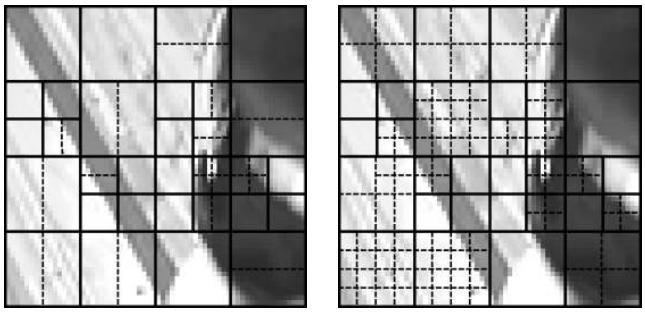
The second post-processing stage is a nonlinear SAO transformation. It was introduced in HEVC to compensate general losses in the decoded CTU. The reconstructed CTU pixel values are modified by adding predetermined offset values. These values are determined by the encoder and transmitted in the bitstream. The affected pixels are chosen by their intensities which determines the nonlinearity.

4 Intra compression in VP9

4.1 Macroblock concept

The concept of macroblock in VP9 is represented by the superblock (Fig. 8). The size of a superblock is 64×64 pixels and its subdivision is pretty much like in HEVC: the partitioning also forms a quad-tree structure. Superblocks can be subdivided down to 4×4 blocks. Each sub-block may be further split on prediction blocks (Fig. 8, *a*) and transform blocks (Fig. 8, *b*). Unlike HEVC, any sub-block can be split on prediction blocks in intra mode. Furthermore rectangular intra-prediction blocks are possible. Intra-prediction in VP9 is still performed on square regions thus rectangular prediction blocks represent two square prediction blocks with the same prediction mode. Giving an analogy to HEVC, prediction splitting 2N×2N, N×N, 2N×N or N×2N is available (Fig. 8, *a*), where 2N×2N is the size of the block being split. It is worth mentioning that 4×4 prediction blocks are determined within corresponding 8×8 block as a group, unlike other prediction sizes when prediction data is stored per each prediction block.

Like in HEVC, a sub-block can be split into transform blocks in a quad-tree structure down to the smallest 4×4 block. The allowed DFT matrix sizes are 32×32, 16×16, 8×8 and 4×4 (Fig. 8, *b*).



a) *b)*

Figure 8: Superblock splitting example with solid lines for block split: a) with prediction splitting depicted as dotted lines; b) with transform splitting depicted as dotted lines

4.2 Intra-prediction modes

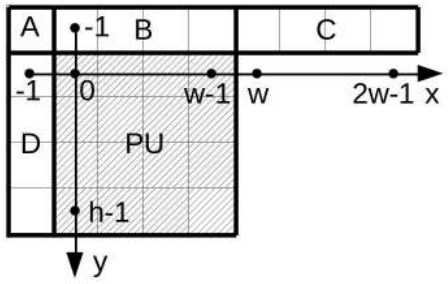


Figure 9: VP9 intra-prediction block

VP9 intra-prediction may involve neighboring pixels from left (set $D = \{p(-1, y)\}$ when $y = 1, \ldots, s - 1$), above (set $B = \{p(x, -1)\}$ when $x = 1, \ldots, s - 1$), above left (set $A = \{p(-1, -1)\}$) and above right (set $C = \{p(x, -1)\}$, $x = s, \ldots, 2 \cdot s - 1$) neighboring pixels (Fig. 9). The availability of those pixels is also determined by block positioning. This is pretty much like in AVC.

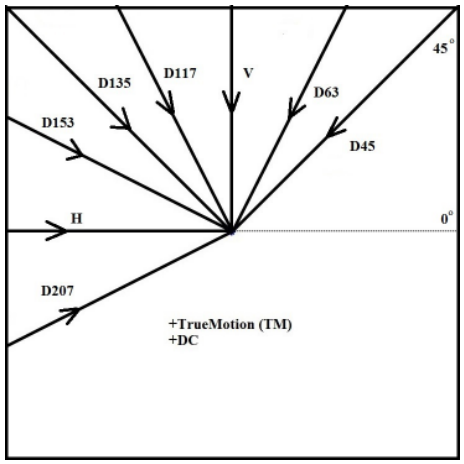


Figure 10: VP9 angular intra-prediction modes

There is a total of ten intra-prediction modes in VP9: DC, True Motion (TM), and eight angular modes (H, V, D207, D153, D135, D117, D63 and D45 on Fig. 10).

To describe intra-prediction modes in VP9 let us do similar assumptions as for HEVC where appropriate. Thus the block being predicted is a pixel matrix $P = \{p(x, y)\}$, where $x = 0, \ldots, w - 1$ and $y = 0, \ldots, h - 1$, with the size $w \times h$ (Fig. 9) and $s = w = h$. An intra-prediction neighbors matrix $R = r(i)$

with $i = -s, \ldots, 2 \cdot s$ is smaller then for HEVC and is formed like:

$$r(i) = \begin{cases} i < 0: & p(-1, -1 - i) \\ i \geqslant 0: & p(i - i, -1) \end{cases}$$

DC prediction in VP9 is much like in HEVC or AVC: each predicted pixel is set equal to the mean value of neighboring pixels:

$$p(x, y) = \frac{1}{2 \cdot s} \cdot \left(\sum_{i=-s}^{-1} r(i) + \sum_{i=1}^{s} r(i) \right)$$

Unlike HEVC, no additional border filtering is performed in DC prediction or any intra-prediction mode.

The True Motion intra-prediction mode in VP9 is an alternative to planar mode in HEVC, but it is far easier. Each pixel value is calculated as follows:

$$p(x, y) = r(-y - 1) + r(x + 1) - r(0)$$

Each directional intra-prediction mode (H, V, D207, D153, D135, D117, D63 and D45) is a linear interpolation of pixel values in corresponding direction. For example, mode D45 is a 45-degree interpolation of pixel values, mode D207 is a 207-degree interpolation of pixel values. All directional intra-prediction modes can be described with the help of two functions:

$$b(i) = \begin{cases} r(i) & if\ i = -s \\ \frac{1}{2} \cdot (r(i-1) + r(i)) & otherwise \end{cases}$$

$$t(i) = \begin{cases} \frac{1}{4} \cdot (3 \cdot r(i) + r(i+1)) & if\ i = -s \\ \frac{1}{4} \cdot (3 \cdot r(i) + r(i-1)) & if\ i = 2 \cdot s \\ \frac{1}{4} \cdot (r(i-1) + 2 \cdot r(i) + r(i+1)) & otherwise \end{cases}$$

Bearing in mind the size of R is $3 \cdot s + 1$, all 8 directional intra-prediction modes can be determined by $2 \cdot s + 1$ known $r(i)$ values and $6 \cdot s$ precalcuated $b(i)$ and $t(i)$ values, which may come out to be one of the intra-prediction optimization techniques for VP9 encoder.

With the above equations, the D45 predicted pixel value can be described as:

$$p(x, y) = \begin{cases} r(2 \cdot s) & if\ x = s - 1\ and\ y = s - 1 \\ t(x + y + 2) & otherwise \end{cases}$$

and the example prediction matrix for an 8×8 block would be:

$$p(x,y) = \begin{bmatrix} t(2) & t(3) & t(4) & t(5) & t(6) & t(7) & t(8) & t(9) \\ t(3) & t(4) & t(5) & t(6) & t(7) & t(8) & t(9) & t(10) \\ t(4) & t(5) & t(6) & t(7) & t(8) & t(9) & t(10) & t(11) \\ t(5) & t(6) & t(7) & t(8) & t(9) & t(10) & t(11) & t(12) \\ t(6) & t(7) & t(8) & t(9) & t(10) & t(11) & t(12) & t(13) \\ t(7) & t(8) & t(9) & t(10) & t(11) & t(12) & t(13) & t(14) \\ t(8) & t(9) & t(10) & t(11) & t(12) & t(13) & t(14) & t(15) \\ t(9) & t(10) & t(11) & t(12) & t(13) & t(14) & t(15) & r(16) \end{bmatrix}$$

The D63 intra-prediction is conducted the following way:

$$p(x,y) = \begin{cases} t\left(\frac{1}{2} \cdot (y+1) + x + 1\right) & for\ odd\ y \\ b\left(\frac{1}{2} \cdot y + x + 2\right) & for\ even\ y \end{cases}$$

and the example prediction matrix for an 8×8 block would be:

$$p(x,y) = \begin{bmatrix} b(2) & b(3) & b(4) & b(5) & b(6) & b(7) & b(8) & b(9) \\ t(2) & t(3) & t(4) & t(5) & t(6) & t(7) & t(8) & t(9) \\ b(3) & b(4) & b(5) & b(6) & b(7) & b(8) & b(9) & b(10) \\ t(3) & t(4) & t(5) & t(6) & t(7) & t(8) & t(9) & t(10) \\ b(4) & b(5) & b(6) & b(7) & b(8) & b(9) & b(10) & b(11) \\ t(4) & t(5) & t(6) & t(7) & t(8) & t(9) & t(10) & t(11) \\ b(5) & b(6) & b(7) & b(8) & b(9) & b(10) & b(11) & b(12) \\ t(5) & t(6) & t(7) & t(8) & t(9) & t(10) & t(11) & t(12) \end{bmatrix}$$

The V intra-prediction mode stands for vertical interpolation direction, when the predicted pixel has the following value:

$$p(x,y) = r(x+1)$$

and the example prediction matrix for an 8×8 block would be:

$$p(x,y) = \begin{bmatrix} r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \\ r(1) & r(2) & r(3) & r(4) & r(5) & r(6) & r(7) & r(8) \end{bmatrix}$$

The D117 intra-prediction is conducted the following way:

$$p(x,y) = \begin{cases} b\left(x - \frac{1}{2} \cdot y\right) & for\ even\ y\ and\ x \geqslant \frac{1}{2} \cdot y \\ t\left(x - \frac{1}{2} \cdot (y+1)\right) & for\ odd\ y\ and\ x \geqslant \frac{1}{2} \cdot (y+1) \\ t\left(2 \cdot x - y\right) & otherwise \end{cases}$$

and the example prediction matrix for an 8×8 block:

$$p(x,y) = \begin{bmatrix} b(0) & b(1) & b(2) & b(3) & b(4) & b(5) & b(6) & b(7) \\ t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) & t(5) & t(6) \\ t(-2) & b(0) & b(1) & b(2) & b(3) & b(4) & b(5) & b(6) \\ t(-3) & t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) & t(5) \\ t(-4) & t(-2) & b(0) & b(1) & b(2) & b(3) & b(4) & b(5) \\ t(-5) & t(-3) & t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) \\ t(-6) & t(-4) & t(-2) & b(0) & b(1) & b(2) & b(3) & b(4) \\ t(-7) & t(-5) & t(-3) & t(-1) & t(0) & t(1) & t(2) & t(3) \end{bmatrix}$$

The D135 intra-prediction mode has the following function system (not finished):

$$p(x,y) = t(x-y)$$

and the example prediction matrix for an 8×8 block:

$$p(x,y) = \begin{bmatrix} t(0) & t(1) & t(2) & t(3) & t(4) & t(5) & t(6) & t(7) \\ t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) & t(5) & t(6) \\ t(-2) & t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) & t(5) \\ t(-3) & t(-2) & t(-1) & t(0) & t(1) & t(2) & t(3) & t(4) \\ t(-4) & t(-3) & t(-2) & t(-1) & t(0) & t(1) & t(2) & t(3) \\ t(-5) & t(-4) & t(-3) & t(-2) & t(-1) & t(0) & t(1) & t(2) \\ t(-6) & t(-5) & t(-4) & t(-3) & t(-2) & t(-1) & t(0) & t(1) \\ t(-7) & t(-6) & t(-5) & t(-4) & t(-3) & t(-2) & t(-1) & t(0) \end{bmatrix}$$

The D153 directional prediction:

$$p(x,y) = \begin{cases} b\left(\frac{1}{2} \cdot x - y\right) & for\ even\ x\ and\ y \geqslant \frac{1}{2} \cdot x \\ t\left(\frac{1}{2} \cdot (x-1) - y\right) & for\ odd\ x\ and\ y \geqslant \frac{1}{2} \cdot (x-1) \\ t\left(x - 2 \cdot y - 1\right) & otherwise \end{cases}$$

with the example 8×8 prediction matrix:

$$p(x,y) = \begin{bmatrix} b(0) & t(0) & t(1) & t(2) & t(3) & t(4) & t(5) & t(6) \\ b(-1) & t(-1) & b(0) & t(0) & t(1) & t(2) & t(3) & t(4) \\ b(-2) & t(-2) & b(-1) & t(-1) & b(0) & t(0) & t(1) & t(2) \\ b(-3) & t(-3) & b(-2) & t(-2) & b(-1) & t(-1) & b(0) & t(0) \\ b(-4) & t(-4) & b(-3) & t(-3) & b(-2) & t(-2) & b(-1) & t(-1) \\ b(-5) & t(-5) & b(-4) & t(-4) & b(-3) & t(-3) & b(-2) & t(-2) \\ b(-6) & t(-6) & b(-5) & t(-5) & b(-4) & t(-4) & b(-3) & t(-3) \\ b(-7) & t(-7) & b(-6) & t(-6) & b(-5) & t(-5) & b(-4) & t(-4) \end{bmatrix}$$

The horizontal intra-prediction mode, depicted as H, is just a row spread of neighboring value:

$$p(x,y) = r(-1-y)$$

with the example 8×8 prediction matrix:

$$p(x,y) = \begin{bmatrix} r(-1) & r(-1) & r(-1) & r(-1) & r(-1) & r(-1) & r(-1) & r(-1) \\ r(-2) & r(-2) & r(-2) & r(-2) & r(-2) & r(-2) & r(-2) & r(-2) \\ r(-3) & r(-3) & r(-3) & r(-3) & r(-3) & r(-3) & r(-3) & r(-3) \\ r(-4) & r(-4) & r(-4) & r(-4) & r(-4) & r(-4) & r(-4) & r(-4) \\ r(-5) & r(-5) & r(-5) & r(-5) & r(-5) & r(-5) & r(-5) & r(-5) \\ r(-6) & r(-6) & r(-6) & r(-6) & r(-6) & r(-6) & r(-6) & r(-6) \\ r(-7) & r(-7) & r(-7) & r(-7) & r(-7) & r(-7) & r(-7) & r(-7) \\ r(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) \end{bmatrix}$$

The D207 intra-prediction mode has the following equation:

$$p(x,y) = \begin{cases} r(-s) & if\ x \geqslant 2 \cdot (s-y) - 1 \\ b(-y - \frac{1}{2} \cdot x - 1) & for\ even\ x \\ t(-y - \frac{1}{2} \cdot (x+1) - 1) & for\ odd\ x \end{cases}$$

with the example 8×8 prediction matrix:

$$p(x,y) = \begin{bmatrix} b(-1) & t(-2) & b(-2) & t(-3) & b(-3) & t(-4) & b(-4) & t(-5) \\ b(-2) & t(-3) & b(-3) & t(-4) & b(-4) & t(-5) & b(-5) & t(-6) \\ b(-3) & t(-4) & b(-4) & t(-5) & b(-5) & t(-6) & b(-6) & t(-7) \\ b(-4) & t(-5) & b(-5) & t(-6) & b(-6) & t(-7) & b(-7) & t(-8) \\ b(-5) & t(-6) & b(-6) & t(-7) & b(-7) & t(-8) & b(-8) & r(-8) \\ b(-6) & t(-7) & b(-7) & t(-8) & b(-8) & r(-8) & r(-8) & r(-8) \\ b(-7) & t(-8) & b(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) \\ b(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) & r(-8) \end{bmatrix}$$

4.3 Transform and quantization

The residuals after subtraction of predicted pixel values are subjected to DFT and quantization. Transform blocks can be 32×32, 16×16, 8×8 or 4×4 pixels. VP9 uses integer approximation of DCT II and DST II for all transform sizes except for 4×4 transform where DST VI is used by classification in [1]. In addition, VP9 introduces support for a new transform type, the Asymmetric Discrete Sine Transform (ADST), which can be used in combination with specific intra-prediction modes. Intra-prediction modes that predict from a left edge can use the 1-D ADST in the horizontal direction, combined with a 1-D DCT in the vertical direction. Similarly, the residual signal resulting from intra-prediction modes that predict from the top edge can employ a vertical 1-D ADST transform combined with a horizontal 1-D DCT transform. Intra-prediction modes that predict from both edges such as the True Motion mode and some diagonal intra-prediction modes, use the 1-D ADST in both horizontal and vertical directions [4].

4.4 Entropy coding

VP9 uses 8-bit arithmetic coding engine from VP8 known as bool-coder. Unlike AVC or HEVC, the probabilities of VP9 bool-coder do not change adaptively within a frame. VP9 makes use of forward context updates through the use of flags in the frame header that signal modifications of the coding contexts at the start of each frame. These probabilities are stored in what is known as a frame context. The decoder maintains four of these contexts, and each frame specifies which one to use in bitstream.

4.5 Post-processing

There is only one possible post-processing stage in VP9: deblock filter. It aims to reduce blockiness artifacts on superblocks filtering vertical edges first, and horizontal edges second. VP9 has 16-, 8-, 4- and 2-pixels wide filters with half filter size on each side of a boundary. VP9 also incorporates a flatness detector in the loop filter that detects flat regions and varies the filter strength and size accordingly.

5 Experimental results

The experimental comparison was carried out on the JCT-VC video sequences, listed in Table 2. They have different resolutions and frame-rates, covering the most usecases possible. Bitrate-PSNR plots are illustrated in Fig. 11–15. The summarized results are present in Table 3.

Table 2: Test video sequence set

Sequence	Resolution, pixels	Frame-rate, Hz	Number of frames
BlowingBubbles	416×240	50	500
BasketballDrill	832×480	50	500
FourPeople	1280×720	60	600
Kimono	1920×1080	24	240
PeopleOnStreet	2560×1600	30	150

For comparison purposes open-source implementations of the reviewed codecs were used. HEVC compression efficiency was measured with the HM Test Model. Verification of coding parameters was done with Elecard HEVC Analyzer. The HM encoder is configured with "constant quantization" mode when all compressed frames has the same quantizer. This configuration eliminates an influence of the rate-control efficiency on the encoder.

Evaluation of VP9 and VP8 compression performance was carried out with the VPX encoder from The WebM Project as it is the only implementation of this standard. The CodecVisa Analyzer was used for verification of compression parameters. The VPX encoder was configured with the "constrained quality" mode and limited quantization parameter to emulate the "constant quantizer" mode.

For AVC evaluation the JM reference encoder and Elecard Stream Analyzer were used. It was also configured to work in the "constant quantizer" mode.

As stated in the overview, HEVC has more intra-prediction modes than VP9. Some angular intra prediction modes in both standards have similar interpolation directions. To estimate a profit of using more prediction directions, we modified the HM encoder implementation to use only 10 intra prediction modes which alternatives are present in VP9. The HM encoder was allowed to use planar (mode 0), DC (mode 1) and 8 angular prediction modes 5, 10, 15, 18, 21, 26, 31 and 34. The alternative VP9 modes are DC, True Motion, D207, H, D153, D135, D117, V, D63 and D45 respectively. This HM modification is depicted as "HM less angles" on Fig. 11–15 and in Tables 3–4.

To bring HEVC even closer to VP9, in addition to the previous modification, we also turned SAO post-processing stage off because it is not present in VP9. This HM configuration is depicted as "HM less angles without SAO" on Fig. 11–15 and in Tables 3–4.

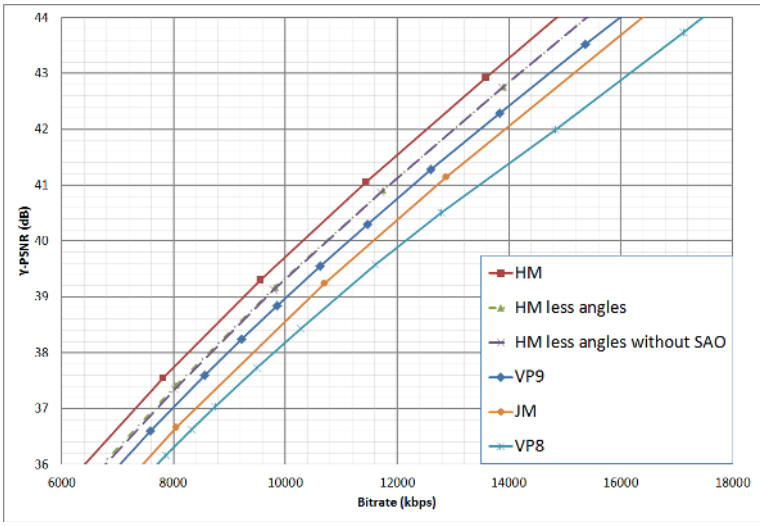


Figure 11: Bitrate-PSNR plot for intra-frame coding on BlowingBubbles sequence

On the BlowingBubbles sequence (Fig. 11) the HM encoder is 20.7% more efficient than the VP9 encoder. Limiting HM intra prediction modes to 10 decreases HM compression efficiency by 11.8%. When we turned SAO off, the HM compression efficiency was further reduced by 2.4%, but still the modified HM encoder is 6.5% more efficient that the VP9 encoder.

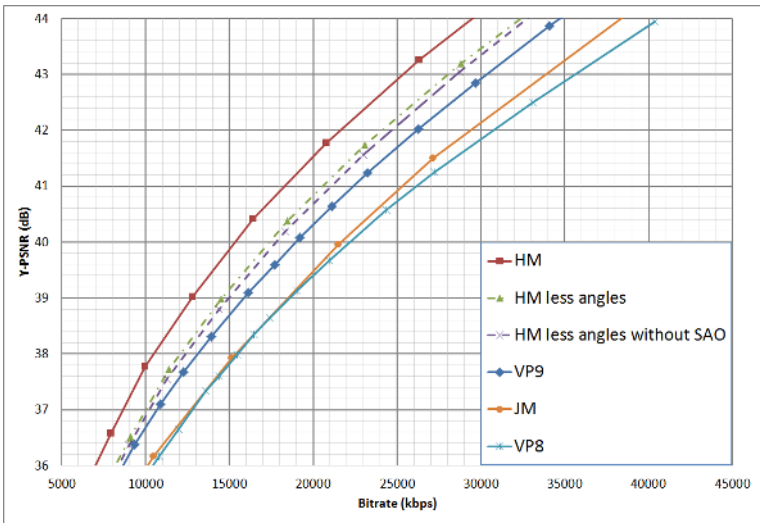


Figure 12: Bitrate-PSNR plot for intra-frame coding on BasketballDrill sequence

On the BasketballDrill sequence (Fig. 12) the HM encoder is 13.9% more efficient than the VP9 encoder. Again limiting HM intra prediction modes,

the bitrate increases by 7%. Turning SAO off make the modified HM encoder 19.6% less efficient that the genuine version, but stil 6.6% more efficient than the VP9 encoder.

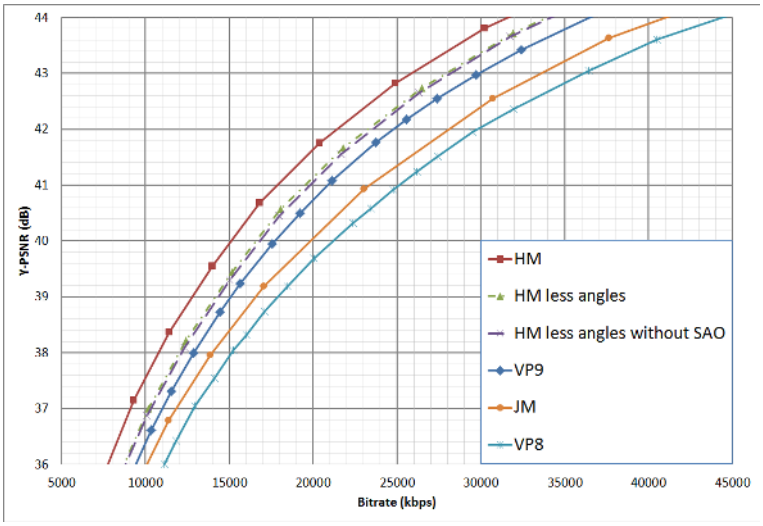


Figure 13: Bitrate-PSNR plot for intra-frame coding on FourPeople sequence

On the FourPeople sequence (Fig. 13) the HM encoder is 16.8% more efficient than the VP9 encoder. Without 25 intra prediction modes the HM encoder is 7.5% more efficient, and without SAO post-processing it still provides 6% lower bitrate compared to VP9.

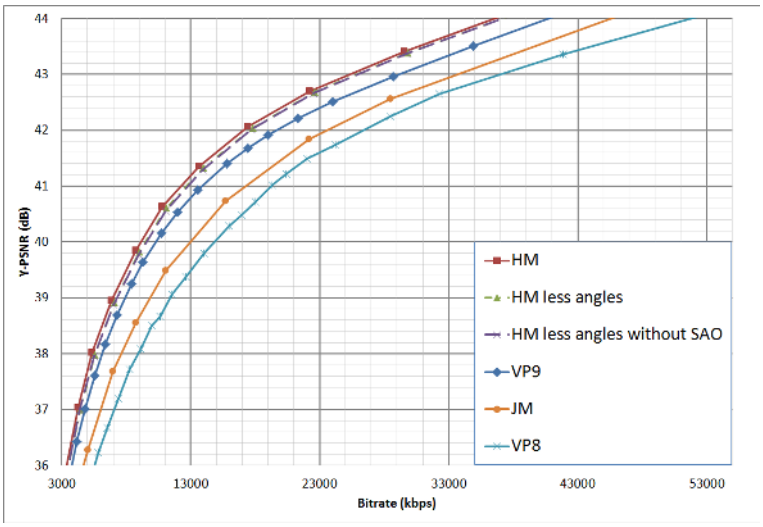


Figure 14: Bitrate-PSNR plot for intra-frame coding on Kimono sequence

A bit different results were achieved on Kimono video sequence. The HM

encoder is 12% more efficient than the VP9 encoder. However HM intra prediction modes limit does not have as much effect as on the other video sequences: bitrate increase is only 2.4%. This is due to the structure of visual data in this sequence. The dominant HM intra prediction modes for this sequence are Planar, DC and vertical mode 26. All this modes are present in the modified HM encoder therefore the impact of intra prediction restrictions is rather low. If SAO is additionally turned off, the modified HM encoder is still 8.8% more efficient compared to VP9.

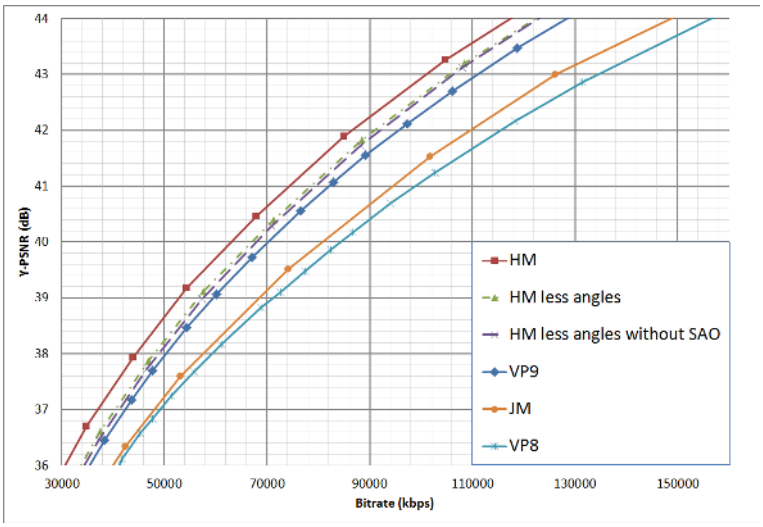


Figure 15: Bitrate-PSNR plot for intra-frame coding on PeopleOnStreet sequence

The results for PeopleOnStreet video sequence are almost similar. The HM encoder provides 14.5% bitrate savings compared to the VP9 encoder. Turning several intra modes off makes it only 6.7% more efficient than VP9. And turning SAO off makes it 4.9% more efficient than VP9.

Table 3: The HM encoder bitrate compared to the VP9 encoder

Sequence \ Codec	HM	HM less angles	HM less angles, SAO off
BlowingBubbles	79,3%	91,1%	93,5%
BasketballDrill	86,1%	93,1%	93,4%
FourPeople	83,2%	92,5%	94,0%
Kimono	88,0%	90,4%	91,2%
PeopleOnStreet	85,5%	93,3%	95,1%
Average	**84,4%**	**92,1%**	**93,4%**

The comparison of the HM encoder bitrate with the VP9 encoder results at the same PSNR quality level is given in Table 3. The main HEVC bitrate savings compared to VP9 are provided by additional intra prediction modes: they comprise 7.7% on average. SAO post-processing stage provides only 1.3% bitrate savings.

Table 4: HM and VP9 encoders bitrate compared to the JM encoder

Sequence \ Codec	HM	VP9
BlowingBubbles	78.16%	92.19%
BasketballDrill	66.83%	85.19%
FourPeople	73.25%	89.02%
Kimono	72.31%	81.94%
PeopleOnStreet	68.65%	81.97%
Average	**71.84%**	**86.06%**

Both HM and VP9 are more efficienct compared to the JM encoder (Table 4) in intra frame compression. But while VP9 provides 14% bitrate decrease for intra compression compared to the JM encoder, the HM encoder provides the whole 28% bitrate decrease at the same visual quality on average. Considering the aim of VP9 was to increase compression efficiency twice compared to VP8, VP9 is about 18% more efficient than VP8. It is also worth mentioning that VP9 provides about 28% bitrate savings compared to the VP8 encoder on the Kimono video sequence. Also it seems to be no significant dependency on resulution of the intra coded video sequence. Compression efficiency is mainly determined by the structure and content of video frame.

6 Conclusion

Both VP9 and HEVC compression standards provide higher compressoin efficiency compared to the current industrial video compression standard AVC. HEVC provides better compression rates than VP9, but VP9 is patent-free and can be used without licensing expenses.

In our experiments we tried to figure out the main reasons for HEVC to be more efficient in intra coding compared to VP9. We showed that more angular intra prediction modes provide the most significant influence on intra compression efficiency (about 7.7% bitrate savings). The HEVC SAO post-processing stage has less impact (about 1.3% bitrate savings). Finally the modified HM encoder with 10 intra prediction modes and without SAO post-processing is still about 7% more efficient compared to the VP9 encoder. We assume this is due to the adaptive HEVC entropy coding. Syntax-adaptive

BAC should be more efficient than frame-adaptive bool-coder, and this topic is a subject for further research.

Further experiments should also be carried out on inter-compression efficiency of both standards.

Acknowledgements. The results were obtained at Tomsk State University of Control Systems and Radioelectronics as part of the complex project 'Provision of multimedia broadcasting services in Internet public networks, based on peer-to-peer network technology and adaptive data streaming' with the financial support of the Ministry of Education and Science of the Russian Federation.

References

[1] V. Britanak, P.C. Yip and K.R. Rao, *Discrete Cosine and Sine Transforms. General Properties, Fast Algorithms and Integer Approximation,* Academic Press, London, 2006.

[2] Cisco. 2013. "Cisco Visual Networking Index: Global Data Traffic Forecast Update, 2012-2017," White Paper, February 2013.

[3] A. Fuldseth, G. Bjontegaard, M. Sadafale, M. Budagavi, *Transform design for HEVC with 16 bit intermediate data representation,* Doc. JCTVC-E243, Geneva, CH, 16-23 March, 2011.

[4] A. Grange, H. Alvestrand, *A VP9 Bitstream Overview (Internet-Draft),* Google, August 2013.

[5] Recommendation H.265: High efficiency video coding, ITU-T, April 2013.

[6] O.G. Ponomarev, M.P. Sharabayko, A.A. Pozdnyakov, Research on video compression methods and algorithms efficiency of H.265/HEVC standard, *Elektrosvyazj,* **3** (2013), 29 - 33.

Received: November 1, 2013