

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

A. Fuldseth
G. Bjontegaard
S. Midtskogen
T. Davies
M. Zanaty
Cisco
October 31, 2016

Thor Video Codec
draft-fuldseth-netvc-thor-03

Abstract

This document provides a high-level description of the Thor video codec. Thor is designed to achieve high compression efficiency with moderate complexity, using the well-known hybrid video coding approach of motion-compensated prediction and transform coding.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definitions	5
2.1. Requirements Language	5
2.2. Terminology	6
3. Block Structure	6
3.1. Super Blocks and Coding Blocks	6
3.2. Special Processing at Frame Boundaries	7
3.3. Transform Blocks	8
3.4. Prediction Blocks	8
4. Intra Prediction	8
5. Inter Prediction	9
5.1. Multiple Reference Frames	9
5.2. Bi-Prediction	10
5.3. Improved chroma prediction	10
5.4. Reordered Frames	10
5.5. Interpolated Reference Frames	10
5.6. Sub-Pixel Interpolation	10
5.6.1. Luma Poly-phase Filter	10
5.6.2. Luma Special Filter Position	12
5.6.3. Chroma Poly-phase Filter	13
5.7. Motion Vector Coding	13
5.7.1. Inter0 and Inter1 Modes	13
5.7.2. Inter2 and Bi-Prediction Modes	15
5.7.3. Motion Vector Direction	16
6. Transforms	16
7. Quantization	16
7.1. Quantization matrices	17
7.1.1. Quantization matrix selection	17
7.1.2. Quantization matrix design	18
8. Loop Filtering	18
8.1. Deblocking	18
8.1.1. Luma deblocking	18
8.1.2. Chroma Deblocking	19
8.2. Constrained Low Pass Filter (CLPF)	20
9. Entropy coding	20
9.1. Overview	20
9.2. Low Level Syntax	21
9.2.1. CB Level	21
9.2.2. PB Level	21
9.2.3. TB Level	22
9.2.4. Super Mode	22
9.2.5. CBP	23
9.2.6. Transform Coefficients	23

10. High Level Syntax	25
10.1. Sequence Header	25
10.2. Frame Header	26
11. IANA Considerations	27
12. Security Considerations	27
13. Normative References	27
Authors' Addresses	27

1. Introduction

This document provides a high-level description of the Thor video codec. Thor is designed to achieve high compression efficiency with moderate complexity, using the well-known hybrid video coding approach of motion-compensated prediction and transform coding.

The Thor video codec is a block-based hybrid video codec similar in structure to widespread standards. The high level encoder and decoder structures are illustrated in Figure 1 and Figure 2 respectively.

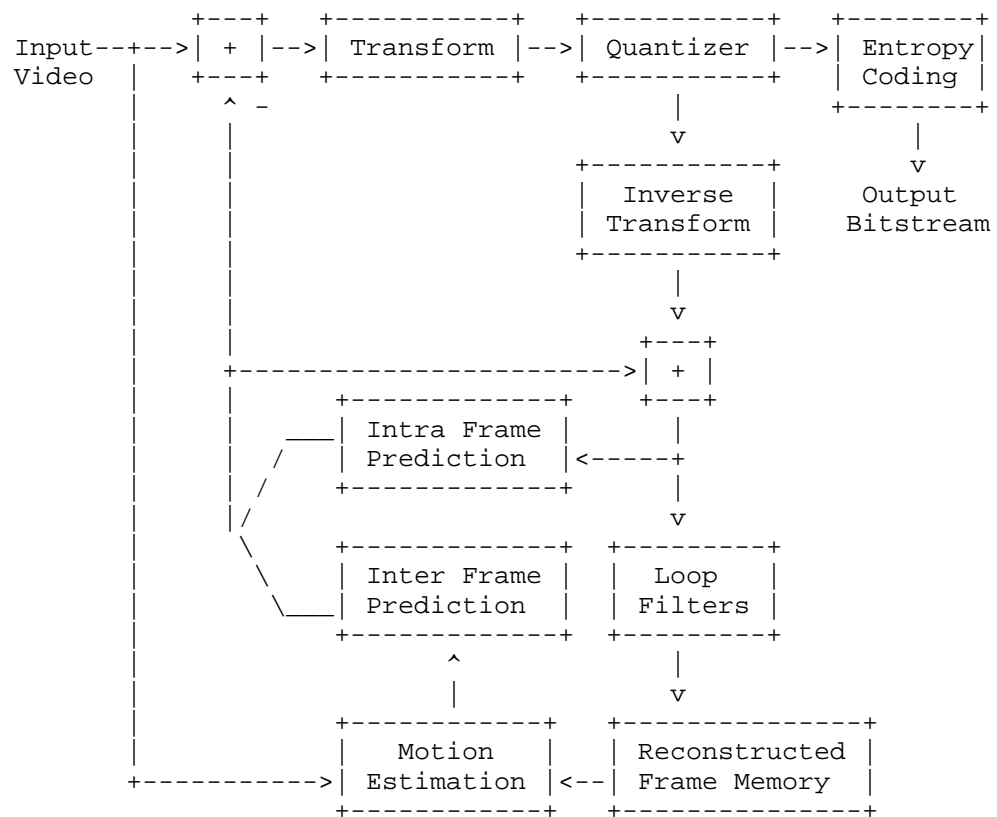


Figure 1: Encoder Structure

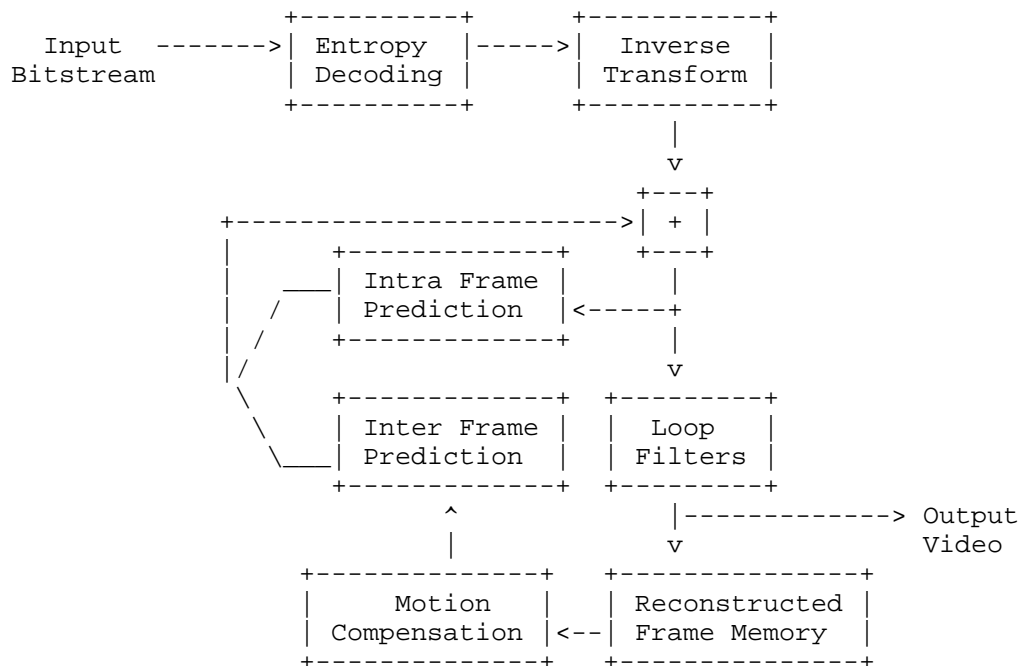


Figure 2: Decoder Structure

The remainder of this document is organized as follows. First, some requirements language and terms are defined. Block structures are described in detail, followed by intra-frame prediction techniques, inter-frame prediction techniques, transforms, quantization, loop filters, entropy coding, and finally high level syntax.

An open source reference implementation is available at github.com/cisco/thor.

2. Definitions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Terminology

This document frequently uses the following terms.

SB: Super Block - 64x64 or 128x128 block (luma pixels) which can be divided into CBs.

CB: Coding Block - Subdivision of a SB, down to 8x8 (luma pixels).

PB: Prediction Block - Subdivision of a CB, into 1, 2 or 4 equal blocks.

TB: Transform Block - Subdivision of a CB, into 1 or 4 equal blocks.

3. Block Structure

3.1. Super Blocks and Coding Blocks

Input frames with bitdepths of 8, 10 or 12 are supported. The internal bitdepth can be 8, 10 or 12 regardless of input bitdepth. The bitdepth of the output frames always follows the input frames. Chroma can be subsampled in both directions (4:2:0) or have full resolution (4:4:4).

Each frame is divided into 64x64 or 128x128 Super Blocks (SB) which are processed in raster-scan order. The SB size is signaled in the sequence header. Each SB can be divided into Coding Blocks (CB) using a quad-tree structure. The smallest allowed CB size is 8x8 luma pixels. The four CBs of a larger block are coded/signaled in the following order; upleft, downleft, upright, and downright.

The following modes are signaled at the CB level:

- o Intra
- o Inter0 (skip): MV index, no residual information
- o Inter1 (merge): MV index, residual information
- o Inter2 (uni-pred): explicit motion information, residual information
- o Inter3 (bi-pred): explicit motion information, residual information

3.2. Special Processing at Frame Boundaries

At frame boundaries some square blocks might not be complete. For example, for 1920x1080 resolutions, the bottom row would consist of rectangular blocks of size 64x56. Rectangular blocks at frame boundaries are handled as follows. For each rectangular block, send one bit to choose between:

- o A rectangular inter0 block and
- o Further split.

For the bottom part of a 1920x1080 frame, this implies the following:

- o For each 64x56 block, transmit one bit to signal a 64x56 inter0 block or a split into two 32x32 blocks and two 32x24 blocks.
- o For each 32x24 block, transmit one bit to signal a 32x24 inter0 block or a split into two 16x16 blocks and two 16x8 blocks.
- o For each 16x8 block, transmit one bit to signal a 16x8 inter0 block or a split into two 8x8 blocks.

Two examples of handling 64x56 blocks at the bottom row of a 1920x1080 frame are shown in Figure 3 and Figure 4 respectively.

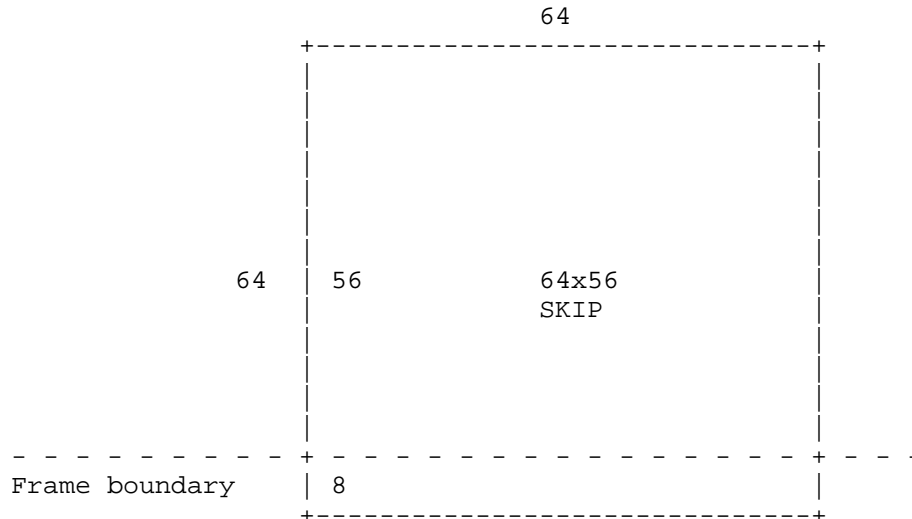


Figure 3: Super block at frame boundary

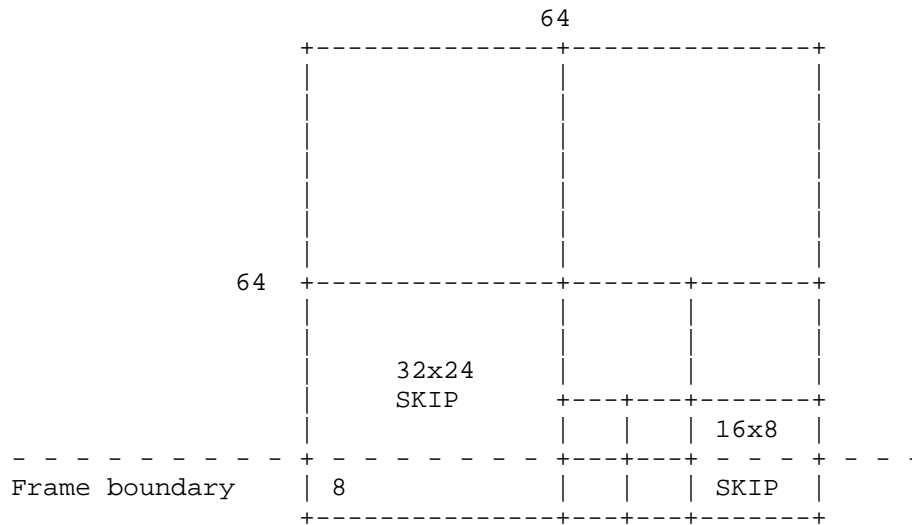


Figure 4: Coding block at frame boundary

3.3. Transform Blocks

A coding block (CB) can be divided into four smaller transform blocks (TBs).

3.4. Prediction Blocks

A coding block (CB) can also be divided into smaller prediction blocks (PBs) for the purpose of motion-compensated prediction. Horizontal, vertical and quad split are used.

4. Intra Prediction

8 intra prediction modes are used:

1. DC
2. Vertical (V)
3. Horizontal (H)
4. Upupright (north-northeast)
5. Upupleft (north-northwest)
6. Upleft (northwest)

7. Upleftleft (west-northwest)

8. Downleftleft (west-southwest)

The definition of DC, vertical, and horizontal modes are straightforward.

The upleft direction is exactly 45 degrees.

The upupright, upupleft, and upleftleft directions are equal to $\arctan(1/2)$ from the horizontal or vertical direction, since they are defined by going one pixel horizontally and two pixels vertically (or vice versa).

For the 5 angular intra modes (i.e. angle different from 90 degrees), the pixels of the neighbor blocks are filtered before they are used for prediction:

$$y(n) = (x(n-1) + 2*x(n) + x(n+1) + 2)/4$$

For the angular intra modes that are not 45 degrees, the prediction sometimes requires sample values at a half-pixel position. These sample values are determined by an additional filter:

$$z(n + 1/2) = (y(n) + y(n+1))/2$$

5. Inter Prediction

5.1. Multiple Reference Frames

Multiple reference frames are currently implemented as follows.

- o Use a sliding-window process to keep the N most recent reconstructed frames in memory. The value of N is signaled in the sequence header.
- o In the frame header, signal which of these frames shall be active for the current frame.
- o For each CB, signal which of the active frames to be used for MC.

Combined with re-ordering, this allows for MPEG-1 style B frames.

A desirable future extension is to allow long-term reference frames in addition to the short-term reference frames defined by the sliding-window process.

5.2. Bi-Prediction

In case of bi-prediction, two reference indices and two motion vectors are signaled per CB. In the current version, PB-split is not allowed in bi-prediction mode. Sub-pixel interpolation is performed for each motion vector/reference index separately before doing an average between the two predicted blocks:

$$p(x,y) = (p0(x,y) + p1(x,y))/2$$

5.3. Improved chroma prediction

If specified in the sequence header, the chroma prediction, both intra and inter, or either, is improved by using the luma reconstruction if certain criteria are met. The process is described in the separate CLPF draft [I-D.midtskogen-netvc-chromapred].

5.4. Reordered Frames

Frames may be transmitted out of order. Reference frames are selected from the sliding window buffer as normal.

5.5. Interpolated Reference Frames

A flag is sent in the sequence header indicating that interpolated reference frames may be used.

If a frame is using an interpolated reference frame, it will be the first reference in the reference list, and will be interpolated from the second and third reference in the list. It is indicated by a reference index of -1 and has a frame number equal to that of the current frame.

The interpolated reference is created by a deterministic process common to the encoder and decoder, and described in the separate IRFVC draft [I-D.davies-netvc-irfvc].

5.6. Sub-Pixel Interpolation

5.6.1. Luma Poly-phase Filter

Inter prediction uses traditional block-based motion compensated prediction with quarter pixel resolution. A separable 6-tap poly-phase filter is the basis method for doing MC with sub-pixel accuracy. The luma filter coefficients are as follows:

When bi-prediction is enabled in the sequence header:

1/4 phase: [2,-10,59,17,-5,1]/64

2/4 phase: [1,-8,39,39,-8,1]/64

3/4 phase: [1,-5,17,59,-10,2]/64

When bi-prediction is disabled in the sequence header:

1/4 phase: [1,-7,55,19,-5,1]/64

2/4 phase: [1,-7,38,38,-7,1]/64

3/4 phase: [1,-5,19,55,-7,1]/64

With reference to Figure 5, a fractional sample value, e.g. $i_{0,0}$ which has a phase of 1/4 in the horizontal dimension and a phase of 1/2 in the vertical dimension is calculated as follows:

$$a_{0,j} = 2*A_{-2,i} - 10*A_{-1,i} + 59*A_{0,i} + 17*A_{1,i} - 5*A_{2,i} + 1*A_{3,i}$$

where $j = -2, \dots, 3$

$$i_{0,0} = (1*a_{0,-2} - 8*a_{0,-1} + 39*a_{0,0} + 39*a_{0,1} - 8*a_{0,2} + 1*a_{0,3} + 2048)/4096$$

The minimum sub-block size is 8x8.

A				A	a	b	c	A
-1,-1				0,-1	0,-1	0,-1	0,-1	1,-1
A				A	a	b	c	A
-1,0				0,0	0,0	0,0	0,0	1,0
d				d	e	f	g	d
-1,0				0,0	0,0	0,0	0,0	1,0
h				h	i	j	k	h
-1,0				0,0	0,0	0,0	0,0	1,0
l				l	m	n	o	l
-1,0				0,0	0,0	0,0	0,0	1,0
A				A	a	b	c	A
-1,1				0,1	0,1	0,1	0,1	1,1

Figure 5: Sub-pixel positions

5.6.2. Luma Special Filter Position

For the fractional pixel position having exactly 2 quarter pixel offsets in each dimension, a non-separable filter is used to calculate the interpolated value. With reference to Figure 5, the center position $j0,0$ is calculated as follows:

$j0,0 =$

$[0 \cdot A_{-1,-1} + 1 \cdot A_{0,-1} + 1 \cdot A_{1,-1} + 0 \cdot A_{2,-1} +$

$1 \cdot A_{-1,0} + 2 \cdot A_{0,0} + 2 \cdot A_{1,0} + 1 \cdot A_{2,0} +$

$1 \cdot A_{-1,1} + 2 \cdot A_{0,1} + 2 \cdot A_{1,1} + 1 \cdot A_{2,1} +$

$$0*A_{-1,2} + 1*A_{0,2} + 1*A_{1,2} + 0*A_{2,2} + 8]/16$$

5.6.3. Chroma Poly-phase Filter

Chroma interpolation is performed with 1/8 pixel resolution using the following poly-phase filter.

1/8 phase: [-2, 58, 10, -2]/64

2/8 phase: [-4, 54, 16, -2]/64

3/8 phase: [-4, 44, 28, -4]/64

4/8 phase: [-4, 36, 36, -4]/64

5/8 phase: [-4, 28, 44, -4]/64

6/8 phase: [-2, 16, 54, -4]/64

7/8 phase: [-2, 10, 58, -2]/64

5.7. Motion Vector Coding

5.7.1. Inter0 and Inter1 Modes

Inter0 and inter1 modes imply signaling of a motion vector index to choose a motion vector from a list of candidate motion vectors with associated reference frame index. A list of motion vector candidates are derived from at most two different neighbor blocks, each having a unique motion vector/reference frame index. Signaling of the motion vector index uses 0 or 1 bit, dependent on the number of unique motion vector candidates. If the chosen neighbor block is coded in bi-prediction mode, the inter0 or inter1 block inherits both motion vectors, both reference indices and the bi-prediction property of the neighbor block.

For block sizes less than 64x64, inter0 has only one motion vector candidate, and its value is always zero.

Which neighbor blocks to use for motion vector candidates depends on the availability of the neighbor blocks (i.e. whether the neighbor blocks have already been coded, belong to the same slice and are not outside the frame boundaries). Four different availabilities, U, UR, L, and LL, are defined as illustrated in Figure 6. If the neighbor block is intra it is considered to be available but with a zero motion vector.

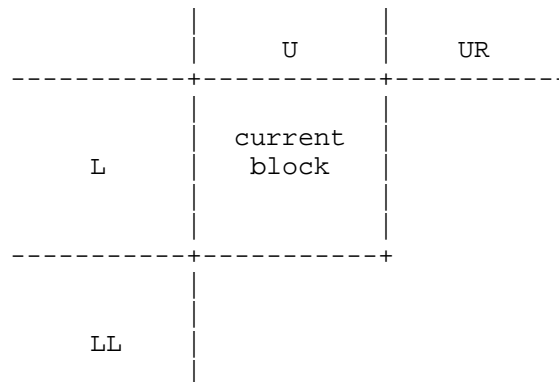


Figure 6: Availability of neighbor blocks

Based on the four availabilities defined above, each of the motion vector candidates is derived from one of the possible neighbor blocks defined in Figure 7.

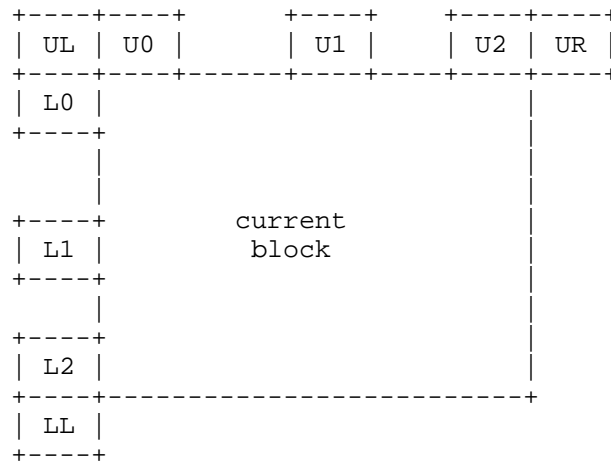


Figure 7: Motion vector candidates

The choice of motion vector candidates depends on the availability of neighbor blocks as shown in Table 1.

U	UR	L	LL	Motion vector candidates
0	0	0	0	zero vector
1	0	0	0	U2, zero vector
0	1	0	0	NA
1	1	0	0	U2, zero vector
0	0	1	0	L2, zero vector
1	0	1	0	U2,L2
0	1	1	0	NA
1	1	1	0	U2,L2
0	0	0	1	NA
1	0	0	1	NA
0	1	0	1	NA
1	1	0	1	NA
0	0	1	1	L2, zero vector
1	0	1	1	U2,L2
0	1	1	1	NA
1	1	1	1	U2,L2

Table 1: Motion vector candidates for different availability of neighbor blocks

5.7.2. Inter2 and Bi-Prediction Modes

Motion vectors are coded using motion vector prediction. The motion vector predictor is defined as the median of the motion vectors from three neighbor blocks. Definition of the motion vector predictor uses the same definition of availability and neighbors as in Figure 6 and Figure 7 respectively. The three vectors used for median filtering depends on the availability of neighbor blocks as shown in Table 2. If the neighbor block is coded in bi-prediction mode, only the first motion vector (in transmission order), MV0, is used as input to the median operator.

U	UR	L	LL	Motion vectors for median filtering
0	0	0	0	3 x zero vector
1	0	0	0	U0,U1,U2
0	1	0	0	NA
1	1	0	0	U0,U2,UR
0	0	1	0	L0,L1,L2
1	0	1	0	UL,U2,L2
0	1	1	0	NA
1	1	1	0	U0,UR,L2,L0
0	0	0	1	NA
1	0	0	1	NA
0	1	0	1	NA
1	1	0	1	NA
0	0	1	1	L0,L2,LL
1	0	1	1	U2,L0,LL
0	1	1	1	NA
1	1	1	1	U0,UR,L0

Table 2: Neighbor blocks used to define motion vector predictor through median filtering

5.7.3. Motion Vector Direction

Motion vectors referring to reference frames later in time than the current frame are stored with their sign reversed, and these reversed values are used for coding and motion vector prediction.

6. Transforms

Transforms are applied at the TB or CB level, implying that transform sizes range from 4x4 to 128x128. The transforms form an embedded structure meaning the transform matrix elements of the smaller transforms can be extracted from the larger transforms.

7. Quantization

For the 32x32, 64x64 and 128x128 transform sizes, only the 16x16 low frequency coefficients are quantized and transmitted.

The 64x64 inverse transform is defined as a 32x32 transform followed by duplicating each output sample into a 2x2 block. The 128x128 inverse transform is defined as a 32x32 transform followed by duplicating each output sample into a 4x4 block.

7.1. Quantization matrices

A flag is transmitted in the sequence header to indicate whether quantization matrices are used. If this flag is true, a 6 bit value `qmtx_offset` is transmitted in the sequence header to indicate matrix strength.

If used, then in dequantization a separate scaling factor is applied to each coefficient, so that the dequantized value of a coefficient `ci` at position `i` is:

$$(ci * d(q) * IW(i,c,s,t,q) + 2^{(k + 5)}) >> (k + 6)$$

Figure 8: Equation 1

where `IW` is the scale factor for coefficient position `i` with size `s`, frame type (inter/inter) `t`, component (Y, Cb or Cr) `c` and quantizer `q`; and `k=k(s,q)` is the dequantization shift. `IW` has scale 64, that is, a weight value of 64 is no different to unweighted dequantization.

7.1.1.1. Quantization matrix selection

The current luma `qp` value `qpY` and the offset value `qmtx_offset` determine a quantisation matrix set by the formula:

$$qmlevel = \max(0, \min(11, ((qpY + qmtx_offset) * 12) / 44))$$

Figure 9: Equation 2

This selects one of the 12 different sets of default quantization matrix, with increasing `qmlevel` indicating increasing flatness.

For a given value of `qmlevel`, different weighting matrices are provided for all combinations of transform block size, type (intra/inter), and component (Y, Cb, Cr). Matrices at low `qmlevel` are flat (constant value 64). Matrices for inter frames have unity DC gain (i.e. value 64 at position 0), whereas those for intra frames are designed such that the inverse weighting matrix has unity energy gain (i.e. normalized sum-squared of the scaling factors is 1).

7.1.2. Quantization matrix design

Further details on the quantization matrix and implementation can be found in the separate QMTX draft [I-D.davies-netvc-qmtx].

8. Loop Filtering

8.1. Deblocking

8.1.1. Luma deblocking

Luma deblocking is performed on an 8x8 grid as follows:

1. For each vertical edge between two 8x8 blocks, calculate the following for each of line 2 and line 5 respectively:

$$d = \text{abs}(a-b) + \text{abs}(c-d),$$

where a and b, are on the left hand side of the block edge and c and d are on the right hand side of the block edge:

a b | c d

2. For each line crossing the vertical edge, perform deblocking if and only if all of the following conditions are true:

- * $d2+d5 < \text{beta}(\text{QP})$

- * The edge is also a transform block edge

- * $\text{abs}(\text{mvx}(\text{left})) > 2$, or $\text{abs}(\text{mvx}(\text{right})) > 2$, or

- $\text{abs}(\text{mvy}(\text{left})) > 2$, or $\text{abs}(\text{mvy}(\text{right})) > 2$, or

- One of the transform blocks on each side of the edge has non-zero coefficients, or

- One of the transform blocks on each side of the edge is coded using intra mode.

3. If deblocking is performed, calculate a delta value as follows:

$$\text{delta} = \text{clip}((18*(c-b) - 6*(d-a) + 16)/32, \text{tc}, -\text{tc}),$$

where tc is a QP-dependent value.

4. Next, modify two pixels on each side of the block edge as follows:

$$a' = a + \text{delta}/2$$

$$b' = b + \text{delta}$$

$$c' = c + \text{delta}$$

$$d' = d + \text{delta}/2$$

5. The same procedure is followed for horizontal block edges.

The relative positions of the samples, a, b, c, d and the motion vectors, MV, are illustrated in Figure 10.

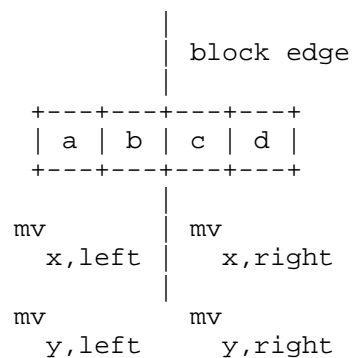


Figure 10: Deblocking filter pixel positions

8.1.2. Chroma Deblocking

Chroma deblocking is performed on a 4x4 grid as follows:

1. Delocking of the edge between two 4x4 blocks is performed if and only if:
 - * The pixels on either side of the block edge belongs to an intra block.
 - * The block edge is also an edge between two transform blocks.
2. If deblocking is performed, calculate a delta value as follows:

$$\text{delta} = \text{clip}((4*(c-b) + (d-a) + 4)/8, \text{tc}, -\text{tc}),$$
 where tc is a QP-dependent value.

3. Next, modify one pixel on each side of the block edge as follows:

$$b' = b + \text{delta}$$

$$c' = c + \text{delta}$$

8.2. Constrained Low Pass Filter (CLPF)

A low-pass filter is applied after the deblocking filter if signaled in the sequence header. It can still be switched off for individual frames in the frame header. Also signaled in the frame header is whether to apply the filter for all qualified 128x128 blocks or to transmit a flag for each such block. A super block does not qualify if it only contains Inter0 (skip) coding block and no signal is transmitted for these blocks.

The filter is described in the separate CLPF draft [I-D.midtskogen-netvc-clpf].

9. Entropy coding

9.1. Overview

The following information is signaled at the sequence level:

- o Sequence header

The following information is signaled at the frame level:

- o Frame header

The following information is signaled at the CB level:

- o Super-mode (mode, split, reference index for uni-prediction)
- o Intra prediction mode
- o PB-split (none, hor, ver, quad)
- o TB-split (none or quad)
- o Reference frame indices for bi-prediction
- o Motion vector candidate index
- o Transform coefficients if TB-split=0

The following information is signaled at the TB level:

- o CBP (8 combinations of CBPY, CBPU, and CBPV)
- o Transform coefficients

The following information is signaled at the PB level:

- o Motion vector differences

9.2. Low Level Syntax

9.2.1. CB Level

super-mode (inter0/split/inter1/inter2-ref0/intra/inter2-ref1/inter2-ref2/inter2-ref3,...)

if (mode == inter0 || mode == inter1)

mv_idx (one of up to 2 motion vector candidates)

else if (mode == INTRA)

intra_mode (one of up to 8 intra modes)

tb_split (NONE or QUAD, coded jointly with CBP for tb_split=NONE)

else if (mode == INTER)

pb_split (NONE, VER, HOR, QUAD)

tb_split_and_cbp (NONE or QUAD and CBP)

else if (mode == BIPRED)

mvd_x0, mvd_y0 (motion vector difference for first vector)

mvd_x1, mvd_y1 (motion vector difference for second vector)

ref_idx0, ref_idx1 (two reference indices)

9.2.2. PB Level

if (mode == INTER2 || mode == BIPRED)

mvd_x, mvd_y (motion vector differences)

9.2.3. TB Level

```

    if (mode != INTER0 and tb_split == 1)

        cbp                                (8 possibilities for CBPY/CBPU/CBPV)

    if (mode != INTER0)

        transform coefficients

```

9.2.4. Super Mode

For each block of size $N \times N$ ($64 \geq N > 8$), the following mutually exclusive events are jointly encoded using a single VLC code as follows (example using 4 reference frames):

If there is no interpolated reference frame:

```

INTER0      1
SPLIT       01
INTER1      001
INTER2-REF0 0001
BIPRED      00001
INTRA       000001
INTER2-REF1 0000001
INTER2-REF2 00000001
INTER2-REF3 00000000

```

If there is an interpolated reference frame:

```

INTER0      1
SPLIT       01
INTER1      001
BIPRED      0001
INTRA       00001
INTER2-REF1 000001
INTER2-REF2 0000001
INTER2-REF3 00000001
INTER2-REF0 00000000

```

If less than 4 reference frames is used, a shorter VLC table is used. If bi-pred is not possible, or split is not possible, they are omitted from the table and shorter codes are used for subsequent elements.

Additionally, depending on information from the blocks to the left and above (meta data and CBP), a different sorting of the events can be used, e.g.:

SPLIT	1
INTER1	01
INTER2-REF0	001
INTER0	0001
INTRA	00001
INTER2-REF1	000001
INTER2-REF2	0000001
INTER2-REF3	00000001
BIPRED	00000000

9.2.5. CBP

Calculate code as follows:

```
if (tb-split == 0)

    N = 4*CBPV + 2*CBPU + CBPY

else

    N = 8
```

Map the value of N to code through a table lookup:

```
code = table[N]
```

where the purpose of the table lookup is the sort the different values of code according to decreasing probability (typically CBPY=1, CBPU=0, CBPV=0 having the highest probability).

Use a different table depending on the values of CBPY in neighbor blocks (left and above).

Encode the value of code using a systematic VLC code.

9.2.6. Transform Coefficients

Transform coefficient coding uses a traditional zig-zag scan pattern to convert a 2D array of quantized transform coefficients, *coeff*, to a 1D array of samples. VLC coding of quantized transform coefficients starts from the low frequency end of the 1D array using two different modes; level-mode and run-mode, starting in level-mode:

- o Level-mode
 - * Encode each coefficient, *coeff*, separately
 - * Each coefficient is encoded by:

- + The absolute value, $\text{level} = \text{abs}(\text{coeff})$, using a VLC code and
 - + If $\text{level} > 0$, the sign bit ($\text{sign}=0$ or $\text{sign}=1$ for $\text{coeff}>0$ and $\text{coeff}<0$ respectively).
 - * If coefficient N is zero, switch to run-mode, starting from coefficient $N+1$.
- o Run-mode
- * For each non-zero coefficient, encode the combined event of:
 1. Length of the zero-run, i.e. the number of zeros since the last non-zero coefficient.
 2. Whether or not $\text{level} = \text{abs}(\text{coeff})$ is greater than 1.
 3. End of block (EOB) indicating that there are no more non-zero coefficients.
 - * Additionally, if $\text{level} = 1$, code the sign bit.
 - * Additionally, if $\text{level} > 1$ define $\text{code} = 2 * (\text{level} - 2) + \text{sign}$,
 - * If the absolute value of coefficient N is larger than 1, switch to level-mode, starting from coefficient $N+1$.

Example

Figure 11 illustrates an example where 16 quantized transform coefficients are encoded.

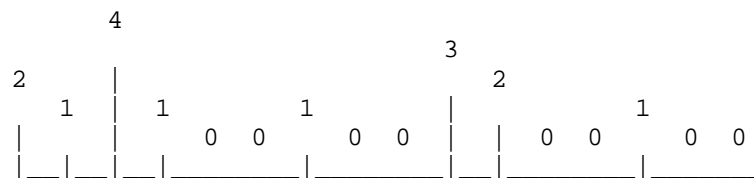


Figure 11: Coefficients to encode

Table 3 shows the mode, VLC number and symbols to be coded for each coefficient.

Index	abs(coeff)	Mode	Encoded symbols
0	2	level-mode	level=2,sign
1	1	level-mode	level=1,sign
2	4	level-mode	level=4,sign
3	1	level-mode	level=1,sign
4	0	level-mode	level=0
5	0	run-mode	
6	1	run-mode	(run=1,level=1)
7	0	run-mode	
8	0	run-mode	
9	3	run-mode	(run=1,level>1), 2*(3-2)+sign
10	2	level-mode	level=2, sign
11	0	level-mode	level=0
12	0	run-mode	
13	1	run-mode	(run=1,level=1)
14	0	run-mode	EOB
15	0	run-mode	

Table 3: Transform coefficient encoding for the example above.

10. High Level Syntax

High level syntax is currently very simple and rudimentary as the primary focus so far has been on compression performance. It is expected to evolve as functionality is added.

10.1. Sequence Header

- o Width - 16 bits
- o Height - 16 bits
- o Enable/disable PB-split - 1 bit
- o SB size - 3 bits
- o Enable/disable TB-split - 1 bit
- o Number of active reference frames (may go into frame header) - 2 bits (max 4)
- o Enable/disable interpolated reference frames - 1 bit
- o Enable/disable delta qp - 1 bit

- o Enable/disable deblocking - 1 bit
- o Constrained low-pass filter (CLPF) enable/disable - 1 bit
- o Enable/disable block context coding - 1 bit
- o Enable/disable bi-prediction - 1 bit
- o Enable/disable quantization matrices - 1 bit
- o If quantization matrices enabled: quantization matrix offset - 6 bits
- o Select 420 or 444 input - 1 bit
- o Number of reordered frames - 4 bits
- o Enable/disable chroma intra prediction from luma - 1 bit
- o Enable/disable chroma inter prediction from luma - 1 bit
- o Internal frame bitdepth (8, 10 or 12 bits) - 2 bits
- o Input video bitdepth (8, 10 or 12 bits) - 2 bits

10.2. Frame Header

- o Frame type - 1 bit
- o QP - 8 bits
- o Identification of active reference frames - num_ref*4 bits
- o Number of intra modes - 4 bits
- o Number of active reference frames - 2 bits
- o Active reference frames - number of active reference frames * 6 bits
- o Frame number - 16 bits
- o If CLPF is enabled in the sequence header: Constrained low-pass filter (CLPF) strength - 2 bits (00 = off, 01 = strength 1, 10 = strength 2, 11 = strength 4)
- o IF CLPF is enabled in the sequence header: Enable/disable CLPF signal for each qualified filter block

11. IANA Considerations

This document has no IANA considerations yet. TBD

12. Security Considerations

This document has no security considerations yet. TBD

13. Normative References

[I-D.davies-netvc-irfvc]

Davies, T., "Interpolated reference frames for video coding", draft-davies-netvc-irfvc-00 (work in progress), October 2015.

[I-D.davies-netvc-qmtx]

Davies, T., "Quantisation matrices for Thor video coding", draft-davies-netvc-qmtx-00 (work in progress), March 2016.

[I-D.midtskogen-netvc-chromapred]

Midtskogen, S., "Improved chroma prediction", draft-midtskogen-netvc-chromapred-02 (work in progress), October 2016.

[I-D.midtskogen-netvc-clpf]

Midtskogen, S., Fuldseth, A., and M. Zanaty, "Constrained Low Pass Filter", draft-midtskogen-netvc-clpf-02 (work in progress), April 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

Authors' Addresses

Arild Fuldseth
Cisco
Lysaker
Norway

Email: arilfuld@cisco.com

Gisle Bjontegaard
Cisco
Lysaker
Norway

Email: gbjonteg@cisco.com

Steinar Midtskogen
Cisco
Lysaker
Norway

Email: stemidts@cisco.com

Thomas Davies
Cisco
London
UK

Email: thdavies@cisco.com

Mo Zanaty
Cisco
RTP,NC
USA

Email: mzanaty@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 3, 2020

T. Daede
Mozilla
A. Norkin
Netflix
I. Brailovski
Amazon Lab126
January 31, 2020

Video Codec Testing and Quality Measurement
draft-ietf-netvc-testing-09

Abstract

This document describes guidelines and procedures for evaluating a video codec. This covers subjective and objective tests, test conditions, and materials used for the test.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 3, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Subjective quality tests	3
2.1. Still Image Pair Comparison	3
2.2. Video Pair Comparison	4
2.3. Mean Opinion Score	4
3. Objective Metrics	5
3.1. Overall PSNR	5
3.2. Frame-averaged PSNR	5
3.3. PSNR-HVS-M	6
3.4. SSIM	6
3.5. Multi-Scale SSIM	6
3.6. CIEDE2000	6
3.7. VMAF	6
4. Comparing and Interpreting Results	7
4.1. Graphing	7
4.2. BD-Rate	7
4.3. Ranges	8
5. Test Sequences	8
5.1. Sources	8
5.2. Test Sets	8
5.2.1. regression-1	9
5.2.2. objective-2-slow	9
5.2.3. objective-2-fast	12
5.2.4. objective-1.1	14
5.2.5. objective-1-fast	17
5.3. Operating Points	19
5.3.1. Common settings	19
5.3.2. High Latency CQP	19
5.3.3. Low Latency CQP	19
5.3.4. Unconstrained High Latency	20
5.3.5. Unconstrained Low Latency	20
6. Automation	20
6.1. Regression tests	21
6.2. Objective performance tests	21
6.3. Periodic tests	22
7. IANA Considerations	22
8. Security Considerations	22
9. Informative References	22
Authors' Addresses	23

1. Introduction

When developing a video codec, changes and additions to the codec need to be decided based on their performance tradeoffs. In addition, measurements are needed to determine when the codec has met its performance goals. This document specifies how the tests are to be carried about to ensure valid comparisons when evaluating changes under consideration. Authors of features or changes should provide the results of the appropriate test when proposing codec modifications.

2. Subjective quality tests

Subjective testing uses human viewers to rate and compare the quality of videos. It is the preferable method of testing video codecs.

Subjective testing results take priority over objective testing results, when available. Subjective testing is recommended especially when taking advantage of psychovisual effects that may not be well represented by objective metrics, or when different objective metrics disagree.

Selection of a testing methodology depends on the feature being tested and the resources available. Test methodologies are presented in order of increasing accuracy and cost.

Testing relies on the resources of participants. If a participant requires a subjective test for a particular feature or improvement, they are responsible for ensuring that resources are available. This ensures that only important tests be done; in particular, the tests that are important to participants.

Subjective tests should use the same operating points as the objective tests.

2.1. Still Image Pair Comparison

A simple way to determine superiority of one compressed image is to visually compare two compressed images, and have the viewer judge which one has a higher quality. For example, this test may be suitable for an intra de-ringing filter, but not for a new inter prediction mode. For this test, the two compressed images should have similar compressed file sizes, with one image being no more than 5% larger than the other. In addition, at least 5 different images should be compared.

Once testing is complete, a p-value can be computed using the binomial test. A significant result should have a resulting p-value less than or equal to 0.5. For example:

```
p_value = binom_test(a,a+b)
```

where a is the number of votes for one video, b is the number of votes for the second video, and `binom_test(x,y)` returns the binomial PMF (probability mass function) with x observed tests, y total tests, and expected probability 0.5.

If ties are allowed to be reported, then the equation is modified:

```
p_value = binom_test(a+floor(t/2),a+b+t)
```

where t is the number of tie votes.

Still image pair comparison is used for rapid comparisons during development – the viewer may be either a developer or user, for example. As the results are only relative, it is effective even with an inconsistent viewing environment. Because this test only uses still images (keyframes), this is only suitable for changes with similar or no effect on inter frames.

2.2. Video Pair Comparison

The still image pair comparison method can be modified to also compare videos. This is necessary when making changes with temporal effects, such as changes to inter-frame prediction. Video pair comparisons follow the same procedure as still images. Videos used for testing should be limited to 10 seconds in length, and can be rewatched an unlimited number of times.

2.3. Mean Opinion Score

A Mean Opinion Score (MOS) viewing test is the preferred method of evaluating the quality. The subjective test should be performed as either consecutively showing the video sequences on one screen or on two screens located side-by-side. The testing procedure should normally follow rules described in [BT500] and be performed with non-expert test subjects. The result of the test will be (depending on the test procedure) mean opinion scores (MOS) or differential mean opinion scores (DMOS). Confidence intervals are also calculated to judge whether the difference between two encodings is statistically significant. In certain cases, a viewing test with expert test subjects can be performed, for example if a test should evaluate technologies with similar performance with respect to a particular artifact (e.g. loop filters or motion prediction). Unlike pair

comparisons, a MOS test requires a consistent testing environment. This means that for large scale or distributed tests, pair comparisons are preferred.

3. Objective Metrics

Objective metrics are used in place of subjective metrics for easy and repeatable experiments. Most objective metrics have been designed to correlate with subjective scores.

The following descriptions give an overview of the operation of each of the metrics. Because implementation details can sometimes vary, the exact implementation is specified in C in the Daala tools repository [DAALA-GIT]. Implementations of metrics must directly support the input's resolution, bit depth, and sampling format.

Unless otherwise specified, all of the metrics described below only apply to the luma plane, individually by frame. When applied to the video, the scores of each frame are averaged to create the final score.

Codecs must output the same resolution, bit depth, and sampling format as the input.

3.1. Overall PSNR

PSNR is a traditional signal quality metric, measured in decibels. It is directly derived from mean square error (MSE), or its square root (RMSE). The formula used is:

$$20 * \log_{10} (\text{MAX} / \text{RMSE})$$

or, equivalently:

$$10 * \log_{10} (\text{MAX}^2 / \text{MSE})$$

where the error is computed over all the pixels in the video, which is the method used in the `dump_psnr.c` reference implementation.

This metric may be applied to both the luma and chroma planes, with all planes reported separately.

3.2. Frame-averaged PSNR

PSNR can also be calculated per-frame, and then the values averaged together. This is reported in the same way as overall PSNR.

3.3. PSNR-HVS-M

The PSNR-HVS [PSNRHVS] metric performs a DCT transform of 8x8 blocks of the image, weights the coefficients, and then calculates the PSNR of those coefficients. Several different sets of weights have been considered. The weights used by the `dump_pnsrhvs.c` tool in the Daala repository have been found to be the best match to real MOS scores.

3.4. SSIM

SSIM (Structural Similarity Image Metric) is a still image quality metric introduced in 2004 [SSIM]. It computes a score for each individual pixel, using a window of neighboring pixels. These scores can then be averaged to produce a global score for the entire image. The original paper produces scores ranging between 0 and 1.

To linearize the metric for BD-Rate computation, the score is converted into a nonlinear decibel scale:

$$-10 * \log_{10} (1 - \text{SSIM})$$

3.5. Multi-Scale SSIM

Multi-Scale SSIM is SSIM extended to multiple window sizes [MSSSIM]. The metric score is converted to decibels in the same way as SSIM.

3.6. CIEDE2000

CIEDE2000 is a metric based on CIEDE color distances [CIEDE2000]. It generates a single score taking into account all three chroma planes. It does not take into consideration any structural similarity or other psychovisual effects.

3.7. VMAF

Video Multi-method Assessment Fusion (VMAF) is a full-reference perceptual video quality metric that aims to approximate human perception of video quality [VMAF]. This metric is focused on quality degradation due to compression and rescaling. VMAF estimates the perceived quality score by computing scores from multiple quality assessment algorithms, and fusing them using a support vector machine (SVM). Currently, three image fidelity metrics and one temporal signal have been chosen as features to the SVM, namely Anti-noise SNR (ANSNR), Detail Loss Measure (DLM), Visual Information Fidelity (VIF), and the mean co-located pixel difference of a frame with respect to the previous frame.

The quality score from VMAF is used directly to calculate BD-Rate, without any conversions.

4. Comparing and Interpreting Results

4.1. Graphing

When displayed on a graph, bitrate is shown on the X axis, and the quality metric is on the Y axis. For publication, the X axis should be linear. The Y axis metric should be plotted in decibels. If the quality metric does not natively report quality in decibels, it should be converted as described in the previous section.

4.2. BD-Rate

The Bjontegaard rate difference, also known as BD-rate, allows the measurement of the bitrate reduction offered by a codec or codec feature, while maintaining the same quality as measured by objective metrics. The rate change is computed as the average percent difference in rate over a range of qualities. Metric score ranges are not static - they are calculated either from a range of bitrates of the reference codec, or from quantizers of a third, anchor codec. Given a reference codec and test codec, BD-rate values are calculated as follows:

- o Rate/distortion points are calculated for the reference and test codec.
 - * At least four points must be computed. These points should be the same quantizers when comparing two versions of the same codec.
 - * Additional points outside of the range should be discarded.
- o The rates are converted into log-rates.
- o A piecewise cubic hermite interpolating polynomial is fit to the points for each codec to produce functions of log-rate in terms of distortion.
- o Metric score ranges are computed:
 - * If comparing two versions of the same codec, the overlap is the intersection of the two curves, bound by the chosen quantizer points.
 - * If comparing dissimilar codecs, a third anchor codec's metric scores at fixed quantizers are used directly as the bounds.

- o The log-rate is numerically integrated over the metric range for each curve, using at least 1000 samples and trapezoidal integration.
- o The resulting integrated log-rates are converted back into linear rate, and then the percent difference is calculated from the reference to the test codec.

4.3. Ranges

For individual feature changes in libaom or libvpx, the overlap BD-Rate method with quantizers 20, 32, 43, and 55 must be used.

For the final evaluation described in [I-D.ietf-netvc-requirements], the quantizers used are 20, 24, 28, 32, 36, 39, 43, 47, 51, and 55.

5. Test Sequences

5.1. Sources

Lossless test clips are preferred for most tests, because the structure of compression artifacts in already-compressed clips may introduce extra noise in the test results. However, a large amount of content on the internet needs to be recompressed at least once, so some sources of this nature are useful. The encoder should run at the same bit depth as the original source. In addition, metrics need to support operation at high bit depth. If one or more codecs in a comparison do not support high bit depth, sources need to be converted once before entering the encoder.

5.2. Test Sets

Sources are divided into several categories to test different scenarios the codec will be required to operate in. For easier comparison, all videos in each set should have the same color subsampling, same resolution, and same number of frames. In addition, all test videos must be publicly available for testing use, to allow for reproducibility of results. All current test sets are available for download [TESTSEQUENCES].

Test sequences should be downloaded in whole. They should not be recreated from the original sources.

Each clip is labeled with its resolution, bit depth, color subsampling, and length.

5.2.1. regression-1

This test set is used for basic regression testing. It contains a very small number of clips.

- o kirlandvga (640x360, 8bit, 4:2:0, 300 frames)
- o FourPeople (1280x720, 8bit, 4:2:0, 60 frames)
- o Narrator (4096x2160, 10bit, 4:2:0, 15 frames)
- o CSGO (1920x1080, 8bit, 4:4:4 60 frames)

5.2.2. objective-2-slow

This test set is a comprehensive test set, grouped by resolution. These test clips were created from originals at [TESTSEQUENCES]. They have been scaled and cropped to match the resolution of their category. This test set requires a codec that supports both 8 and 10 bit video.

4096x2160, 4:2:0, 60 frames:

- o Netflix_BarScene_4096x2160_60fps_10bit_420_60f
- o Netflix_BoxingPractice_4096x2160_60fps_10bit_420_60f
- o Netflix_Dancers_4096x2160_60fps_10bit_420_60f
- o Netflix_Narrator_4096x2160_60fps_10bit_420_60f
- o Netflix_RitualDance_4096x2160_60fps_10bit_420_60f
- o Netflix_ToddlerFountain_4096x2160_60fps_10bit_420_60f
- o Netflix_WindAndNature_4096x2160_60fps_10bit_420_60f
- o street_hdr_amazon_2160p

1920x1080, 4:2:0, 60 frames:

- o aspen_1080p_60f
- o crowd_run_1080p50_60f
- o ducks_take_off_1080p50_60f
- o guitar_hdr_amazon_1080p

- o life_1080p30_60f
- o Netflix_Aerial_1920x1080_60fps_8bit_420_60f
- o Netflix_Boat_1920x1080_60fps_8bit_420_60f
- o Netflix_Crosswalk_1920x1080_60fps_8bit_420_60f
- o Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f
- o Netflix_PierSeaside_1920x1080_60fps_8bit_420_60f
- o Netflix_SquareAndTimelapse_1920x1080_60fps_8bit_420_60f
- o Netflix_TunnelFlag_1920x1080_60fps_8bit_420_60f
- o old_town_cross_1080p50_60f
- o pan_hdr_amazon_1080p
- o park_joy_1080p50_60f
- o pedestrian_area_1080p25_60f
- o rush_field_cuts_1080p_60f
- o rush_hour_1080p25_60f
- o seaplane_hdr_amazon_1080p
- o station2_1080p25_60f
- o touchdown_pass_1080p_60f

1280x720, 4:2:0, 120 frames:

- o boat_hdr_amazon_720p
- o dark720p_120f
- o FourPeople_1280x720_60_120f
- o gipsrestat720p_120f
- o Johnny_1280x720_60_120f
- o KristenAndSara_1280x720_60_120f

- o Netflix_DinnerScene_1280x720_60fps_8bit_420_120f
- o Netflix_DrivingPOV_1280x720_60fps_8bit_420_120f
- o Netflix_FoodMarket2_1280x720_60fps_8bit_420_120f
- o Netflix_RollerCoaster_1280x720_60fps_8bit_420_120f
- o Netflix_Tango_1280x720_60fps_8bit_420_120f
- o rain_hdr_amazon_720p
- o vidyo1_720p_60fps_120f
- o vidyo3_720p_60fps_120f
- o vidyo4_720p_60fps_120f

640x360, 4:2:0, 120 frames:

- o blue_sky_360p_120f
- o controlled_burn_640x360_120f
- o desktop2360p_120f
- o kirland360p_120f
- o mmstationary360p_120f
- o niklas360p_120f
- o rain2_hdr_amazon_360p
- o red_kayak_360p_120f
- o riverbed_360p25_120f
- o shields2_640x360_120f
- o snow_mnt_640x360_120f
- o speed_bag_640x360_120f
- o stockholm_640x360_120f
- o tacomanarrows360p_120f

- o thaloundeskg360p_120f
 - o water_hdr_amazon_360p
- 426x240, 4:2:0, 120 frames:

- o bqfree_240p_120f
- o bqhighway_240p_120f
- o bqzoom_240p_120f
- o chairlift_240p_120f
- o dirtbike_240p_120f
- o mozzoom_240p_120f

1920x1080, 4:4:4 or 4:2:0, 60 frames:

- o CSGO_60f.y4m
- o DOTA2_60f_420.y4m
- o MINECRAFT_60f_420.y4m
- o STARCRAFT_60f_420.y4m
- o EuroTruckSimulator2_60f.y4m
- o Hearthstone_60f.y4m
- o wikipedia_420.y4m
- o pvq_slideshow.y4m

5.2.3. objective-2-fast

This test set is a strict subset of objective-2-slow. It is designed for faster runtime. This test set requires compiling with high bit depth support.

1920x1080, 4:2:0, 60 frames:

- o aspen_1080p_60f
- o ducks_take_off_1080p50_60f

- o life_1080p30_60f
- o Netflix_Aerial_1920x1080_60fps_8bit_420_60f
- o Netflix_Boat_1920x1080_60fps_8bit_420_60f
- o Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f
- o Netflix_PierSeaside_1920x1080_60fps_8bit_420_60f
- o Netflix_SquareAndTimelapse_1920x1080_60fps_8bit_420_60f
- o Netflix_TunnelFlag_1920x1080_60fps_8bit_420_60f
- o rush_hour_1080p25_60f
- o seaplane_hdr_amazon_1080p
- o touchdown_pass_1080p_60f

1280x720, 4:2:0, 120 frames:

- o boat_hdr_amazon_720p
- o dark720p_120f
- o gipsrestat720p_120f
- o KristenAndSara_1280x720_60_120f
- o Netflix_DrivingPOV_1280x720_60fps_8bit_420_60f
- o Netflix_RollerCoaster_1280x720_60fps_8bit_420_60f
- o vidyo1_720p_60fps_120f
- o vidyo4_720p_60fps_120f

640x360, 4:2:0, 120 frames:

- o blue_sky_360p_120f
- o controlled_burn_640x360_120f
- o kirland360p_120f
- o niklas360p_120f

- o rain2_hdr_amazon_360p
- o red_kayak_360p_120f
- o riverbed_360p25_120f
- o shields2_640x360_120f
- o speed_bag_640x360_120f
- o thaloundesgmtg360p_120f

426x240, 4:2:0, 120 frames:

- o bqfree_240p_120f
- o bqzoom_240p_120f
- o dirtbike_240p_120f

1290x1080, 4:2:0, 60 frames:

- o DOTA2_60f_420.y4m
- o MINECRAFT_60f_420.y4m
- o STARCRAFT_60f_420.y4m
- o wikipedia_420.y4m

5.2.4. objective-1.1

This test set is an old version of objective-2-slow.

4096x2160, 10bit, 4:2:0, 60 frames:

- o Aerial (start frame 600)
- o BarScene (start frame 120)
- o Boat (start frame 0)
- o BoxingPractice (start frame 0)
- o Crosswalk (start frame 0)
- o Dancers (start frame 120)

- o FoodMarket
- o Narrator
- o PierSeaside
- o RitualDance
- o SquareAndTimelapse
- o ToddlerFountain (start frame 120)
- o TunnelFlag
- o WindAndNature (start frame 120)

1920x1080, 8bit, 4:4:4, 60 frames:

- o CSGO
- o DOTA2
- o EuroTruckSimulator2
- o Hearthstone
- o MINECRAFT
- o STARCRAFT
- o wikipedia
- o pvq_slideshow

1920x1080, 8bit, 4:2:0, 60 frames:

- o ducks_take_off
- o life
- o aspen
- o crowd_run
- o old_town_cross
- o park_joy

- o pedestrian_area
- o rush_field_cuts
- o rush_hour
- o station2
- o touchdown_pass

1280x720, 8bit, 4:2:0, 60 frames:

- o Netflix_FoodMarket2
- o Netflix_Tango
- o DrivingPOV (start frame 120)
- o DinnerScene (start frame 120)
- o RollerCoaster (start frame 600)
- o FourPeople
- o Johnny
- o KristenAndSara
- o vidyo1
- o vidyo3
- o vidyo4
- o dark720p
- o gipsreemotion720p
- o gipsrestat720p
- o controlled_burn
- o stockholm
- o speed_bag
- o snow_mnt

- o shields

640x360, 8bit, 4:2:0, 60 frames:

- o red_kayak
- o blue_sky
- o riverbed
- o thaloundeskmvgvga
- o kirlandvga
- o tacomanarrowsvga
- o tacomascmvvga
- o desktop2360p
- o mmmovingvga
- o mmstationaryvga
- o niklasvga

5.2.5. objective-1-fast

This is an old version of objective-2-fast.

1920x1080, 8bit, 4:2:0, 60 frames:

- o Aerial (start frame 600)
- o Boat (start frame 0)
- o Crosswalk (start frame 0)
- o FoodMarket
- o PierSeaside
- o SquareAndTimelapse
- o TunnelFlag

1920x1080, 8bit, 4:2:0, 60 frames:

- o CSGO
- o EuroTruckSimulator2
- o MINECRAFT

- o wikipedia

1920x1080, 8bit, 4:2:0, 60 frames:

- o ducks_take_off
- o aspen
- o old_town_cross
- o pedestrian_area
- o rush_hour
- o touchdown_pass

1280x720, 8bit, 4:2:0, 60 frames:

- o Netflix_FoodMarket2
- o DrivingPOV (start frame 120)
- o RollerCoaster (start frame 600)
- o Johnny
- o vidyo1
- o vidyo4
- o gipsrecmotion720p
- o speed_bag
- o shields

640x360, 8bit, 4:2:0, 60 frames:

- o red_kayak
- o riverbed

- o kirlandvga
- o tacomascmvvga
- o mmmovingvga
- o niklasvga

5.3. Operating Points

Four operating modes are defined. High latency is intended for on demand streaming, one-to-many live streaming, and stored video. Low latency is intended for videoconferencing and remote access. Both of these modes come in CQP (constant quantizer parameter) and unconstrained variants. When testing still image sets, such as subset1, high latency CQP mode should be used.

5.3.1. Common settings

Encoders should be configured to their best settings when being compared against each other:

- o av1: -codec=av1 -ivf -frame-parallel=0 -tile-columns=0 -cpu-used=0 -threads=1

5.3.2. High Latency CQP

High Latency CQP is used for evaluating incremental changes to a codec. This method is well suited to compare codecs with similar coding tools. It allows codec features with intrinsic frame delay.

- o daala: -v=x -b 2
- o vp9: -end-usage=q -cq-level=x -lag-in-frames=25 -auto-alt-ref=2
- o av1: -end-usage=q -cq-level=x -auto-alt-ref=2

5.3.3. Low Latency CQP

Low Latency CQP is used for evaluating incremental changes to a codec. This method is well suited to compare codecs with similar coding tools. It requires the codec to be set for zero intrinsic frame delay.

- o daala: -v=x
- o av1: -end-usage=q -cq-level=x -lag-in-frames=0

5.3.4. Unconstrained High Latency

The encoder should be run at the best quality mode available, using the mode that will provide the best quality per bitrate (VBR or constant quality mode). Lookahead and/or two-pass are allowed, if supported. One parameter is provided to adjust bitrate, but the units are arbitrary. Example configurations follow:

- o x264: -crf=x
- o x265: -crf=x
- o daala: -v=x -b 2
- o av1: -end-usage=q -cq-level=x -lag-in-frames=25 -auto-alt-ref=2

5.3.5. Unconstrained Low Latency

The encoder should be run at the best quality mode available, using the mode that will provide the best quality per bitrate (VBR or constant quality mode), but no frame delay, buffering, or lookahead is allowed. One parameter is provided to adjust bitrate, but the units are arbitrary. Example configurations follow:

- o x264: -crf=x -tune zerolatency
- o x265: -crf=x -tune zerolatency
- o daala: -v=x
- o av1: -end-usage=q -cq-level=x -lag-in-frames=0

6. Automation

Frequent objective comparisons are extremely beneficial while developing a new codec. Several tools exist in order to automate the process of objective comparisons. The Compare-Codecs tool allows BD-rate curves to be generated for a wide variety of codecs [COMPARECODECS]. The Daala source repository contains a set of scripts that can be used to automate the various metrics used. In addition, these scripts can be run automatically utilizing distributed computers for fast results, with rd_tool [RD_TOOL]. This tool can be run via a web interface called AreWeCompressedYet [AWCY], or locally.

Because of computational constraints, several levels of testing are specified.

6.1. Regression tests

Regression tests run on a small number of short sequences - regression-test-1. The regression tests should include a number of various test conditions. The purpose of regression tests is to ensure bug fixes (and similar patches) do not negatively affect the performance. The anchor in regression tests is the previous revision of the codec in source control. Regression tests are run on both high and low latency CQP modes

6.2. Objective performance tests

Changes that are expected to affect the quality of encode or bitstream should run an objective performance test. The performance tests should be run on a wider number of sequences. The following data should be reported:

- o Identifying information for the encoder used, such as the git commit hash.
- o Command line options to the encoder, configure script, and anything else necessary to replicate the experiment.
- o The name of the test set run (objective-1-fast)
- o For both high and low latency CQP modes, and for each objective metric:
 - * The BD-Rate score, in percent, for each clip.
 - * The average of all BD-Rate scores, equally weighted, for each resolution category in the test set.
 - * The average of all BD-Rate scores for all videos in all categories.

Normally, the encoder should always be run at the slowest, highest quality speed setting (cpu-used=0 in the case of AV1 and VP9). However, in the case of computation time, both the reference and changed encoder can be built with some options disabled. For AV1, -disable-ext_partition and -disable-ext_partition_types can be passed to the configure script to substantially speed up encoding, but the usage of these options must be reported in the test results.

6.3. Periodic tests

Periodic tests are run on a wide range of bitrates in order to gauge progress over time, as well as detect potential regressions missed by other tests.

7. IANA Considerations

This document does not require any IANA actions.

8. Security Considerations

This document describes the methodologies and procedures for qualitative testing, therefore does not itself have implications for network of decoder security.

9. Informative References

- [AWCY] Xiph.Org, "Are We Compressed Yet?", 2016, <<https://arewecompressedyet.com/>>.
- [BT500] ITU-R, "Recommendation ITU-R BT.500-13", 2012, <https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-13-201201-I!!PDF-E.pdf>.
- [CIEDE2000] Yang, Y., Ming, J., and N. Yu, "Color Image Quality Assessment Based on CIEDE2000", 2012, <<http://dx.doi.org/10.1155/2012/273723>>.
- [COMPARECODECS] Alvestrand, H., "Compare Codecs", 2015, <<http://compare-codecs.appspot.com/>>.
- [DAALA-GIT] Xiph.Org, "Daala Git Repository", 2015, <<http://git.xiph.org/?p=daala.git;a=summary>>.
- [I-D.ietf-netvc-requirements] Filippov, A., Norkin, A., and j. jose.roberto.alvarez@huawei.com, "Video Codec Requirements and Evaluation Methodology", draft-ietf-netvc-requirements-10 (work in progress), November 2019.
- [MSSSIM] Wang, Z., Simoncelli, E., and A. Bovik, "Multi-Scale Structural Similarity for Image Quality Assessment", n.d., <<http://www.cns.nyu.edu/~zwang/files/papers/msssim.pdf>>.

- [PSNRHVS] Egiazarian, K., Astola, J., Ponomarenko, N., Lukin, V., Battisti, F., and M. Carli, "A New Full-Reference Quality Metrics Based on HVS", 2002.
- [RD_TOOL] Xiph.Org, "rd_tool", 2016,
<https://github.com/tdaede/rd_tool>.
- [SSIM] Wang, Z., Bovik, A., Sheikh, H., and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", 2004,
<<http://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf>>.
- [TESTSEQUENCES] Daede, T., "Test Sets", n.d.,
<<https://people.xiph.org/~tdaede/sets/>>.
- [VMAF] Aaron, A., Li, Z., Manohara, M., Lin, J., Wu, E., and C. Kuo, "VMAF - Video Multi-Method Assessment Fusion", 2015,
<<https://github.com/Netflix/vmaf>>.

Authors' Addresses

Thomas Daede
Mozilla

Email: tdaede@mozilla.com

Andrey Norkin
Netflix

Email: anorkin@netflix.com

Ilya Brailovskiy
Amazon Lab126

Email: brailovs@lab126.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2018

S. Midtskogen
Cisco
J. Valin
Mozilla
October 26, 2017

Constrained Directional Enhancement Filter
draft-midtskogen-netvc-cdef-00

Abstract

This document describes a constrained directional enhancement filter for use as a loop filter in the Thor video codec.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 29, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Definitions	2
2.1. Requirements Language	2
2.2. Terminology	3
3. Direction search	3
4. Filtering Process	8
5. Signalling	16
6. Results	16
7. IANA Considerations	20
8. Security Considerations	21
9. Acknowledgements	21
10. References	21
10.1. Normative References	21
10.2. Informative References	21
Authors' Addresses	22

1. Introduction

Modern video coding standards such as Thor [I-D.fuldseth-netvc-thor] include in-loop filters which correct artifacts introduced in the encoding process. Thor includes a deblocking filter which corrects artifacts introduced by the block based nature of the encoding process. In addition, Thor introduced the constrained low-pass filter (CLPF [I-D.midtskogen-netvc-clpf]), which compensates for ringing artifacts not corrected by the deblocking filter, and it offers a very favourable complexity/compression trade-off. Similarly, the Daala codec has a deringing filter [I-D.valin-netvc-deringing]. CLPF and the Daala deringing filter have been shown to have additive effects, but rather than running these two filters in cascade after the deblocking filter, they can be combined into a single filter taking advantage of their similarities and reducing the total complexity, giving what we call the constrained directional enhancement filter (CDEF), which will be described in this document. This merged filter offers better compression objectively than CLPF or the Daala deringing filter alone, as well as significantly improved subjective quality, at the cost of somewhat higher complexity than CLPF.

2. Definitions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Terminology

The filter works by dividing the frame to be filtered into filter blocks (FB's) of 64x64 pixels. This size is fixed regardless of the coding block (CB) size which can range from 8x8 to 128x128. Different FB's may have different filter parameters.

If the frame can't fit a whole number of FB's, the FB's at the right and bottom edges are clipped to fit. For instance, if the frame resolution is 1920x1080, the size of the FB's at the bottom of the frame becomes 64x56.

FB's that contain only skipped CB's are never filtered. A CB is skipped when it contains no coded residual.

The frame is further divided into direction blocks (DB) of 8x8 pixels, and all DB's to be filtered are associated with a direction and a variance, both calculated by the encoder and decoder from the reconstructed, deblocked frame. The direction is computed to match the edges and patterns within the DB, and the variance is a measure of the contrast.

CDEF is a non-separable non-linear 12-tap filter and the taps are located within a 5x5 area centered around the pixel to be filtered. One DB is filtered at a time, and the locations of the taps depend on the direction associated with the DB. Furthermore, the taps are divided into two groups: the primary taps and the secondary taps. The primary taps are associated with a primary strength (S'), and the secondary taps are associated with a secondary strength (S''). The primary and secondary strengths can differ.

3. Direction search

The search operates on the reconstructed pixels, just after the deblocking is applied. Since those pixels are available to the decoder, no signalling is required for the directions. The direction search operates on 8x8 blocks (DB's), which is fine enough to adequately handle non-straight edges, while being large enough to reliably estimate directions when applied to a quantized image. Having a constant direction over an 8x8 region also makes vectorization of the filter easier.

For each block we want to determine the direction that best matches the pattern in the block. This is done by minimizing the sum of squared differences (SSD) between the quantized block and a perfectly directional block. A perfectly directional block is a block for which each line along a certain direction has a constant value. For

each direction, we assign a line number k to each pixel, as shown in the following figures:

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

Figure 1: Line number k for pixels following direction 0

0	0	1	1	2	2	3	3
1	1	2	2	3	3	4	4
2	2	3	3	4	4	5	5
3	3	4	4	5	5	6	6
4	4	5	5	6	6	7	7
5	5	6	6	7	7	8	8
6	6	7	7	8	8	9	9
7	7	8	8	9	9	10	10

Figure 2: Line number k for pixels following direction 1

0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7

Figure 3: Line number k for pixels following direction 2

3	3	2	2	1	1	0	0
4	4	3	3	2	2	1	1
5	5	4	4	3	3	2	2
6	6	5	5	4	4	3	3
7	7	6	6	5	5	4	4
8	8	7	7	6	6	5	5
9	9	8	8	7	7	6	6
10	10	9	9	8	8	7	7

Figure 4: Line number k for pixels following direction 3

7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
11	10	9	8	7	6	5	4
12	11	10	9	8	7	6	5
13	12	11	10	9	8	7	6
14	13	12	11	10	9	8	7

Figure 5: Line number k for pixels following direction 4

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
9	8	7	6	5	4	3	2
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
10	9	8	7	6	5	4	3

Figure 6: Line number k for pixels following direction 5

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

Figure 7: Line number k for pixels following direction 6

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10

Figure 8: Line number k for pixels following direction 7

The direction and variance of a DB are calculated by the following algorithm:

```

Initialise all variables to zero
for d = 0 to 7 do
  for i = 0 to 7 do
    for j = 0 to 7 do
      L <- line_table[d][i][j]
      parial[d][L] <- parial[d][L] + (pixel[i][j] - 128)
      count[d][L] <- count[d][L] + 1
    end for
  end for
  for L = 0 to 14 do
    if count[d][L] > 0 then
      s[d] <- s[d] + parial[d][L]^2 * 840 / count[d][L]
    end if
  end for
end for
for d = 0 to 7 do
  if s[d] > s[best_d] then
    best_d <- d
  end if
end for
direction <- best_d
variance <- s[best_d] - s[(best_d + 4) mod 8]

```

Figure 9: Direction search algorithm

Functionally equivalent algebraic simplifications are possible, but they are not shown here for clarity.

4. Filtering Process

CDEF is based on a non-linear low-pass filter designed to remove coding artifacts without blurring sharp edges. This is achieved by selecting taps based on the identified direction, but also by preventing excessive blurring when the filter is applied across an edge. The latter is achieved through the use of a non-linear low-pass filter that deemphasizes taps that differ too much from the pixel being filtered. This filter can be expressed as:

$$y(i,j) = \text{round}(x(i,j) + g(\frac{\sqrt{\sum w'_{m,n} * f(x(i,j)-x(m,m), S', D)}}{\sqrt{\sum w''_{m,n} * f(x(i,j)-x(m,m), S'', D)}}))$$

Figure 10: Non-linear filter

where w' and w'' are the weights associated with the primary and secondary taps respectively, S' and S'' are the primary and secondary strengths, and f , g and the damping value D will be described below.

The function f constrains the difference between the pixel to be filtered and a neighbouring pixel. It takes as arguments the difference d , the strength S and the damping value D :

$$f(d,S,D) = \begin{cases} \min(d, \max(0, S - \text{floor}(d/(2^D - \text{floor}(\log_2(S)))))), & d \geq 0 \\ \max(d, \min(0, \text{floor}(d/(2^D - \text{floor}(\log_2(S)))) - S)), & d < 0 \end{cases}$$

Figure 11: The constrain function

The function restricts the difference to a maximum range defined by the strength, then further restricts large differences depending on the damping value. The constrain function can be visualised as follows

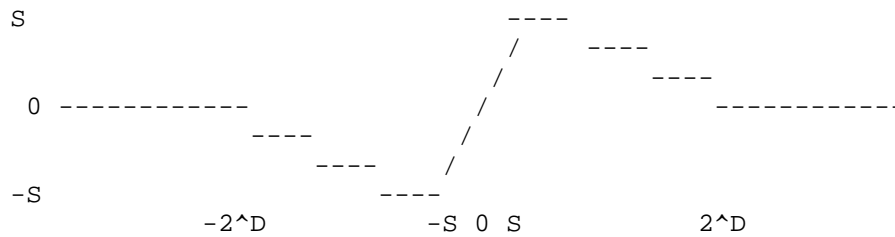


Figure 12: Graph 1

The function is anti-symmetric around $d = 0$ and can be expressed by the following pseudo C code:

```
sign(x) = x >= 0 ? 1 : -1
f(d,S,D) = sign(d)*min(abs(d), max(0, S-(abs(d) >> (D-floor(log2(S))))))
```

Figure 13: The constrain function in pseudo C

The function $g(d)$ is defined as:

$$g(d) = \text{clip}(d, \min_{m,n}(x(i,j) - x(m,n)), \max_{m,n}(x(i,j) - x(m,n)))$$

Figure 14: The clip function

which ensures that the filtered pixel never can attain a value higher than or lower than any of the pixels associated with the filter taps. This has to be done because sum of the weights of the primary and secondary exceeds unity and we want to avoid overcompensation and retain the low-pass quality of the filter.

The direction found in the direction search determines which filter taps to use from the 5x5 area centered around the pixel to be filtered. The primary taps with weights w' are given below for each direction:

```
+---+---+---+---+---+
|   |   |   |   | a |
+---+---+---+---+---+
|   |   |   | b |   |
+---+---+---+---+---+
|   |   | x |   |   |
+---+---+---+---+---+
|   | b |   |   |   |
+---+---+---+---+---+
| a |   |   |   |   |
+---+---+---+---+---+
```

Figure 15: Primary taps (w') for direction 0

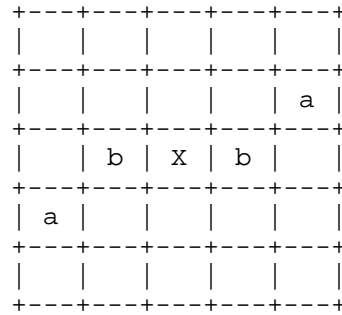


Figure 16: Primary taps (w') for direction 1

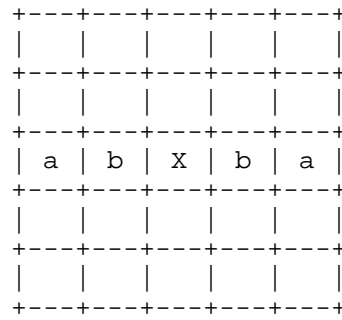


Figure 17: Primary taps (w') for direction 2

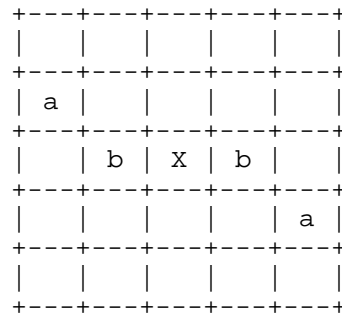


Figure 18: Primary taps (w') for direction 3

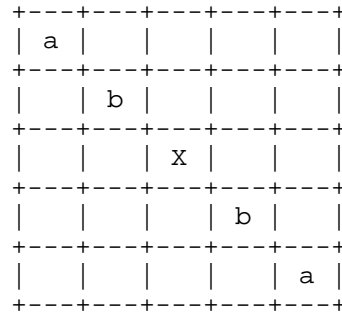


Figure 19: Primary taps (w') for direction 4

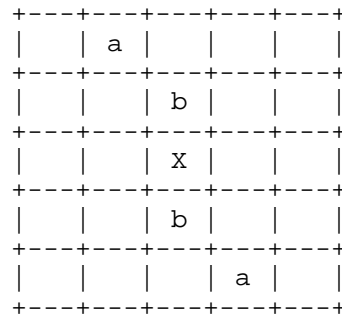


Figure 20: Primary taps (w') for direction 5

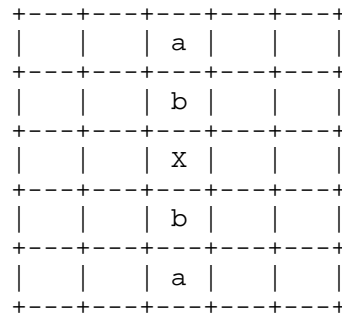


Figure 21: Primary taps (w') for direction 6

+	---	+	---	+	---	+	---	+	---	+
					a					
+	---	+	---	+	---	+	---	+	---	+
					b					
+	---	+	---	+	---	+	---	+	---	+
	a		b		X		b		a	
+	---	+	---	+	---	+	---	+	---	+
					b					
+	---	+	---	+	---	+	---	+	---	+
					a					
+	---	+	---	+	---	+	---	+	---	+

Figure 22: Primary taps (w') for direction 7

The values of a and b alternate depending on the strength. For even strengths, $a = 2/16$ and $b = 4/16$. For odd strengths, $a = b = 3/16$. The secondary taps are as follows:

+	---	+	---	+	---	+	---	+	---	+
					a					
+	---	+	---	+	---	+	---	+	---	+
	a				b					
+	---	+	---	+	---	+	---	+	---	+
			b		X		b			
+	---	+	---	+	---	+	---	+	---	+
					b				a	
+	---	+	---	+	---	+	---	+	---	+
			a							
+	---	+	---	+	---	+	---	+	---	+

Figure 23: Secondary taps (w'') for direction 0 and 4

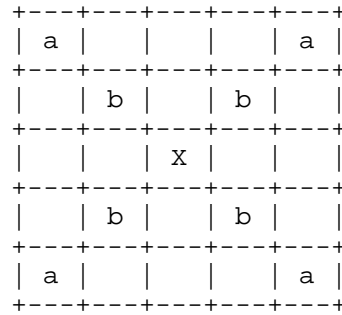


Figure 24: Secondary taps (w'') for direction 1 and 5

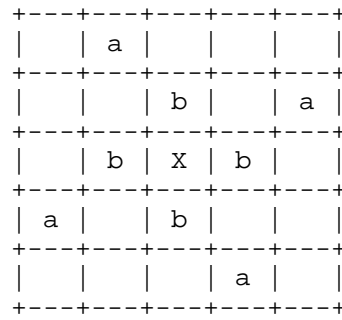


Figure 25: Secondary taps (w'') for direction 2 and 6

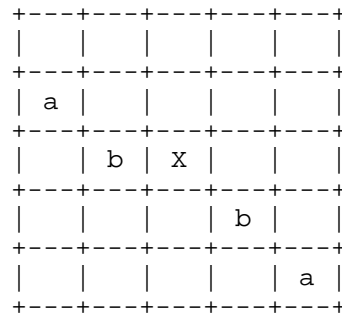


Figure 26: Secondary taps (w'') for direction 3 and 7

For the secondary taps, $a = 1/16$ and $b = 2/16$. Unlike the primary taps, there is no alteration.

The strengths S' and S'' and damping value D must be set high enough to smooth out coding artifacts, but low enough to avoid blurring important details in the image. For 8-bit content S' can have integer values between 0 and 15, and S'' can be 0, 1, 2 or 4. D can be set to 3, 4, 5 or 6 for luma, and the damping value for chroma is always one less. The damping value shall never be lower than the $\log_2(S)$ to ensure that the shift value used to compute $2^{(D - \text{floor}(\log_2(S)))}$ in the constrain function never becomes negative. For instance, if for chroma $S' = 15$ and the luma damping is 3, the chroma damping shall also be 3 (and not 2) because $\text{floor}(\log_2(S')) = 3$.

For higher bit depths (more than 8 bits), S' and S'' are scaled according to the extra bit depth, and D is offset accordingly. For example, 12-bit content can have S' values of 0, 16, 32, ..., 240, and the valid D values are 7, 8, 9 and 10. The weight alteration for the primary taps, which depends on whether the strength is odd or even, are preserved, so for 12-bit content strengths of 16, 48, 80, etc are still considered "odd", and 32, 64, 96, etc are still considered "even".

Picking an optimal damping value is less critical for compression gains than picking the optimal strengths. S' and S'' are chosen independently for luma and chroma.

The primary strength S' is adjusted for luma using a variance v (see the algorithm given in the previous section) for the 8x8 block (DB) as follows:

```
S'_adj =
| floor((S'*(4+min(floor(log2(floor(v/65536)),12))+8)/16), v >= 2^10
|
| 0, otherwise
```

Figure 27: Luma strength adjustment

This adjustment is not applied for chroma, nor for the secondary strength S'' . The adjustment reduces the smoothing for blocks without a clear directional pattern.

5. Signalling

Some CDEF parameters are signaled at the frame level, and some parameters may be signaled at the FB level. The following is signaled at the frame level: the damping D (2 bit), the number of bits used for FB signaling (0-3, 2 bits), and a list of 1, 2, 4 or 8 presets. One preset contains the luma primary strength (4 bits), the chroma primary strength (4 bits), the luma secondary strength (2 bits), the chroma secondary strength, a luma skip condition bit, and a chroma skip condition bit (a total of 14 bits per preset). The filtering is applied one FB at a time. For each FB, the 0 - 3 bits are read to indicate the preset that will be used for this FB. The filter parameters are only coded for FB's that are not completely skipped. Such skipped FB's have CDEF disabled. Similarly, any skipped CB within a FB has filtering disabled unless the skip condition bit is set for that FB.

Since the skip condition flag would be redundant in the case when both the primary and secondary filter strengths are 0, this combination has a special meaning. In that case, the block shall be filtered with a primary filter strength equal to 19 and a secondary filter strength equal to 7. The skip condition flag is still to be regarded as 1.

When the chroma subsampling differs horizontally and vertically, e.g. 4:2:2 video, the filter is disabled for chroma, and the chroma primary strength, the chroma skip condition flag and the chroma secondary strength are not signaled.

6. Results

CDEF has been tested in Thor and AV1 codecs using the Are We Compressed Yet [AWCY] online testing tool and the objective-1-fast test set [I-D.daede-netvc-testing]. The tests were run using different encoder configurations: in high and low latency configurations for three different complexity configurations. The git SHA used for Thor was b5e5cc5 [Thor-git] and for AV1 it was e200b28 [AV1-git].

The filter is more effective for low latency configurations than high latency configuration, and also more effective for low complexity configurations. This makes the filter particularly suited for real-time videoconferencing when low transmission delay is required and expensive compression tools can't be afforded. For encoders requiring a very low complexity, however, CLPF [I-D.midtskogen-netvc-clpf] may still be an attractive alternative.

The tables below show the Bjontegaard Delta Rate (BDR [BDR]) by different metrics, roughly corresponding to bitrate reductions in percent, achieved by CDEF on top of the deblocking filter only (i.e. CLPF always disabled).

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-6.1689	-10.4772	-11.2394	-4.1280	-7.6027	-6.1057	-10.3280

Figure 28: BDR gains in Thor for the low compexity, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-4.0168	-6.3353	-6.6232	-1.6408	-5.3347	-2.9643	-6.3557

Figure 29: BDR gains in Thor for the low compexity, high latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-4.8637	-7.8556	-8.0799	-2.6514	-5.5668	-4.0526	-7.6489

Figure 30: BDR gains in Thor for the medium compexity, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-3.9115	-5.1303	-4.9574	-1.6244	-5.1654	-2.9807	-5.3456

Figure 31: BDR gains in Thor for the medium compexity, high latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-3.1898	-5.2852	-5.4605	-1.3447	-3.3103	-2.2294	-5.1828

Figure 32: BDR gains in Thor for the high efficiency, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-2.2629	-2.7290	-2.5596	-0.4865	-2.7491	-1.3874	-3.1324

Figure 33: BDR gains in Thor for the high efficiency, high latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-3.6819	-3.2943	-4.3394	-2.4961	-4.1543	-3.0463	-4.5402

Figure 34: BDR gains in AV1 for the low complexity, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-1.9320	-2.4224	-3.6913	-0.8598	-1.9586	-1.1803	-2.8803

Figure 35: BDR gains in AV1 for the high efficiency, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-1.0813	-2.1425	-2.7425	-0.1487	-1.1106	-0.4353	-2.1103

Figure 36: BDR gains in AV1 for the high efficiency, high latency configuration

Experiments running both CDEF and the existing CLPF have shown to give only small gains over running CDEF alone, and running both adds a risk of excessive smoothing, so CDEF should be considered a replacement for CLPF, possibly except for encoders with very strict compute budget. Subjective tests of videos encoded in a high efficiency configuration have shown a preference for CDEF for five out of five sequences in the test set. However, the preference was only statistically significant for the low delay configuration. Objectively, CDEF also gives gains when it replaces CLPF as shown in the tables below. The subjective gains appear to be significantly larger. Results are only shown for Thor, as CLPF was not maintained in AV1 during the recent development.

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.8304	-4.0167	-3.6906	-0.7987	-1.3478	-1.1405	-2.1609

Figure 37: BDR over CLPF gains in Thor for the low complexity, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.9475	-2.8048	-2.4094	-0.7117	-0.9714	-0.7862	-1.8283

Figure 38: BDR gains over CLPF in Thor for the low complexity, high latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.8168	-3.5619	-3.4433	-0.7391	-1.1946	-1.0011	-1.9809

Figure 39: BDR gains over CLPF in Thor for the medium compexity, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.7453	-2.6455	-2.5650	-0.4544	-0.7912	-0.4843	-1.6164

Figure 40: BDR gains over CLPF in Thor for the medium compexity, high latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.5777	-2.6286	-2.3601	-0.5300	-1.0664	-0.8435	-1.5601

Figure 41: BDR gains over CLPF in Thor for the high efficiency, low latency configuration

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.4942	-1.6534	-1.5278	-0.4858	-0.9091	-0.7584	-1.0541

Figure 42: BDR gains over CLPF in Thor for the high efficiency, high latency configuration

7. IANA Considerations

This document has no IANA considerations yet. TBD

8. Security Considerations

This document has no security considerations yet. TBD

9. Acknowledgements

The authors would like to thank Thomas Daede for organizing the subjective test.

10. References

10.1. Normative References

- [I-D.daede-netvc-testing]
Daede, T. and J. Jack, "Video Codec Testing and Quality Measurement", draft-daede-netvc-testing-02 (work in progress), October 2015.
- [I-D.fuldseth-netvc-thor]
Fuldseth, A., Bjontegaard, G., Midtskogen, S., Davies, T., and M. Zanaty, "Thor Video Codec", draft-fuldseth-netvc-thor-03 (work in progress), October 2016.
- [I-D.midtskogen-netvc-clpf]
Midtskogen, S., Fuldseth, A., and M. Zanaty, "Constrained Low Pass Filter", draft-midtskogen-netvc-clpf-04 (work in progress), March 2017.
- [I-D.valin-netvc-deringing]
Valin, J., "Directional Deringing Filter", draft-valin-netvc-deringing-01 (work in progress), March 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

10.2. Informative References

- [AV1-git] AOMedia, "AV1 codebase", 2017, <<https://aomedia.googlesource.com/aom>>.
- [AWCY] Xiph.Org, "Are We Compressed Yet?", 2017, <<https://arewecompressedyet.com/>>.
- [BDR] Bjontegaard, G., "Calculation of average PSNR differences between RD-curves", ITU-T SG16 Q6 VCEG-M33 , April 2001.

[Thor-git]
Cisco, "Thor codebase", 2017,
<<https://github.com/cisco/thor>>.

Authors' Addresses

Steinar Midtskogen
Cisco
Lysaker
Norway

Email: stemidts@cisco.com

Jean-Marc Valin
Mozilla
Mountain View
USA

Email: jmvalin@jmvalin.ca

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

J. Samuelsson
P. Hermansson
Divideon
July 2, 2018

The xvc video codec
draft-samuelsson-netvc-xvc-01

Abstract

This document presents a high-level technical description of the xvc video codec. The xvc video codec is a novel video codec that was released in its first version in September 2017. This document provides some technical information as well as a discussion section around how the xvc codec meets the objectives of the NETVC Working Group. The document also includes results comparing the second version of xvc with AV1 and HM for three different test cases; All-Intra, single pass Random-Access, and multi-pass Random-Access. Results are also presented for the full xvc codec relative to the royalty-free baseline profile of xvc.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Technical overview of the xvc video codec	3
2.1. Block structure	4
2.2. Intra prediction	4
2.3. Inter prediction	5
2.4. Transforms	5
2.5. Quantization	5
2.6. Entropy coding	6
2.7. In-loop filtering	6
2.8. Restriction flags	6
2.9. Version handling	10
2.10. Temporal scalability and parallel decoding	11
2.11. Parallel encoding	12
2.12. Baseline profile	12
3. The xvc reference software	12
4. Results	13
4.1. All-Intra	14
4.2. Single pass Random-Access	14
4.3. Multi-pass Random-Access	15
4.4. Full xvc relative to baseline xvc for single pass Random-Access	15
4.5. Full xvc relative to baseline xvc for multi-pass Random-Access	16
4.6. Computational complexity	16
5. Discussion	16
6. IANA Considerations	17
7. Security Considerations	18
8. Informative References	18
Authors' Addresses	19

1. Introduction

In September 2017, the new video compression format xvc was released, with source code publicly available at xvc.io [xvcio]. The xvc codec is a software-defined video codec with conformance for bitstreams, decoders, and encoders, defined with reference to the publicly available reference software. The first version of xvc has been developed by Divideon but the source code of the reference software has been made publicly available with a desire to invite to broad usage and continuous development. The developers of the xvc codec

strongly believe in open, transparent and collaborative processes for technical development, and the purpose of this contribution is to provide information about the xvc project and to present xvc as a candidate for the Internet Video Codec.

The xvc codec was developed with a desire to make efficient compression methods available to a global market under as open, transparent and fair conditions as possible. From the start, the ambition has been to construct a codec, which is easy to deploy, taking into account both technical and non-technical considerations. A deeper analysis of such considerations can be found in the xvc introduction white paper [xvcWp], which provides additional background information related to xvc. In summary, it can be said that one of the most central properties of the xvc codec is that it is not a frozen codec and there is no desire to make it a frozen codec. Instead, conformance is defined with respect to the current official version of the xvc reference software. New versions of the software may be (and is intended to be) released, with addition and/or removal of technologies as deemed feasible. This makes it possible to let the codec evolve over time and ensure that the xvc codec only make use of functionality that is available for licensing under the xvc license [xvcLicense].

Version 2.0 of xvc was released in July, 2018. The new version provides improved compression efficiency compared to xvc 1.0 and includes a royalty-free baseline profile. The software for xvc 2.0 is available under dual-licensing; an LGPL license and a commercial license.

2. Technical overview of the xvc video codec

The xvc codec has been developed completely from scratch and is built primarily using technology that has been investigated in MPEG and VCEG, for example during the HEVC [HEVC] standardization and in exploratory activities after the finalization of HEVC. One of the most fundamental technical properties of the xvc codec is that the different tools (or processing steps) of the codec, have been isolated within separate conditional clauses, controlled by information (restriction flags) included in the bitstream. This makes the xvc codec extremely flexible and it makes it possible to turn off individual processing steps during run-time, if deemed necessary. This design principle is inspired by the proposal in JCTVC-X0034 [Wenger16].

As a summary of the technology in xvc, it can be said that xvc is a block-based hybrid (inter/intra) codec that operates on raw YUV pictures and compresses them to a NAL (network abstraction layer) unit structured bitstream. Each picture in a video sequence is

divided into rectangular blocks of samples, which are predicted from samples in the same picture (intra prediction) or samples in previously coded pictures (inter prediction). Residuals are transformed using non-square transforms and the coded symbols are compressed using a context-adaptive binary arithmetic coder. Block boundaries are filtered using a deblocking filter. The xvc codec supports bit-depths of 8, 10 and 12 bits per sample and chroma formats 4:2:0, 4:2:2, 4:4:4 and monochrome.

2.1. Block structure

A picture that is to be encoded with xvc is divided into blocks of 64x64 pixels, called CTUs. A CTU can either be coded in its full size, or it can be split using a binary split, resulting in two coding units or a quad-tree split, resulting in four coding units. These coding units can be further split into smaller blocks of samples all the way down to coding units of size 4x4 (2x2 for chroma). The splits of a CTU gives rise to a coding unit tree where prediction and transform is performed on the leaf of the trees. Two important differences compared to HEVC can be noted here:

1. In xvc there is no distinction between coding units, prediction units and transform units.
2. In xvc, transform blocks can be non-square.

In the default configuration of xvc, the intra pictures use one coding unit tree for luma and a different coding unit tree for chroma. For inter pictures, a single coding unit tree is used for all color components.

2.2. Intra prediction

There are 67 intra prediction modes in xvc, representing DC prediction, planar prediction and 65 different angular directions. Intra prediction uses previously coded samples to the left and/or above the current block. Reference samples are filtered before being used for prediction. For vertical, horizontal and DC prediction there is also a post filter applied to smooth out the edge to the neighboring block before applying the transform. There is a scheme for estimating which modes are most probable to be used for the next block and through using this scheme the code-words for indicating one of these "most probable modes" can be made shorter than the code-words of other modes. Both chroma components of a block are predicted with the same intra prediction mode, but the chroma prediction mode does not have to be the same as the luma prediction mode. There is also a mode in which the chroma samples are predicted from the luma samples using a linear model.

2.3. Inter prediction

Inter prediction is performed using either uni-prediction (where a single reference picture is used) or bi-prediction (where two reference pictures are used). The xvc codec supports translational motions and affine motions which is applied with different motion vectors for each 4x4 block of a coding unit. The syntax for signaling inter prediction contains shortcuts for indicating that a candidate motion vector is used (merge mode) with a special case for when there is no residual (skip mode). Motion interpolation filters of length up to 8 are used and applied with quarter-pel resolution for luma. In xvc there is a specific inter prediction mode that accounts for local changes between neighboring samples in the current picture and neighboring samples in the reference picture. This mode gives for example good prediction when the light level of a scene changes over time.

2.4. Transforms

On the encoder side, residual data can be derived from taking the original samples of a block and subtract the predicted samples value of the block. This residual can then be forward transformed to result in a set of coefficients that are quantized and signaled in the bitstream. On the decoder side a corresponding (but inverse) process is applied so that coefficients are inverse-quantized, then inverse-transformed and then added to the predicted sample values. In the first version of xvc, a separable DCT-like transform was used where width and height do not need to be of equal length. The size of the transform is always the same as the size of the coding unit. For transforms of size 64, high frequency components are zeroed out. The second version of xvc includes additional transforms that are evaluated and selected based on Rate-Distortion optimization. Regardless of the size of the coding unit, transform coefficients are always coded in subblocks of size 4x4 to enable efficient signaling of subblocks without coefficients.

2.5. Quantization

Transform coefficients in xvc are quantized based on a quantization parameter (QP). Different QP can be used for different CTUs, and the second version of xvc includes support for more efficient signaling of which QP is used for different CTUs. The xvc codec also includes a scheme called sign hiding which makes it possible to derive the sign of one coefficient (instead of signaling it) by rounding the values of the coefficients of a subblock to match the hiding criteria.

2.6. Entropy coding

A Context Adaptive Binary Arithmetic Coder (CABAC) is used in xvc. It resembles the entropy coding method in AVC and HEVC. When a symbol has been coded, the state of the context is updated in order to adjust the probabilities (i.e. the cost) for signaling the same symbol when the same conditions occur again in the same picture (i.e. when the same context is used).

2.7. In-loop filtering

The xvc codec includes a single in-loop filter, which is a deblocking filter, applied on a 4x4 grid of the picture, but only when certain conditions are met. For intra pictures, an edge is filtered if and only if the samples on the different sides of the edge belongs to different coding units. The filtering operation modifies up to two samples on each side of the edge, depending on the local characteristics of the edge.

2.8. Restriction flags

An xvc bitstream starts with a segment header which provides information about the properties of the coded video such as width, height, bitdepth, chroma format etc. The segment header also contains flags that indicate for each individual tool of the codec whether the tool is used in this bitstream or not. These flags correspond to the fundamental design principle in the xvc codec which is to isolate each individual feature (coding tool) within conditional clauses. The conditional clauses are evaluated during run time which means that the decoder is capable of executing both options of each conditional clause. This makes it easy to evaluate the impact of a specific tool in terms of compression and in terms of complexity, without having to compile different versions of the software for each evaluated combination of tools.

In the xvc software, the restriction flags are used instead of #define clauses for different tools. This ensures that the whole codebase is exercised during compilation and makes it easy to verify that different combinations of tools turned on and off work well together. The only macro-defined constant in the xvc software is XVC_HIGH_BITDEPTH, which controls if internal bitdepths above 8 are supported. The first version of xvc contained 62 restriction flags that applies to different areas of the codec. The second version of xvc includes another 14 restriction flags corresponding to new coding tools. An additional restriction flag has been added in order to make it possible to disable the use of the DST transform without disabling the use of 4x4 intra blocks altogether.

The following 25 restriction flags are set to 0 in the default configuration of xvc. The corresponding coding tools are used both in the royalty-free profile and in the full xvc codec.

- o `disable_intra_ref_padding`
- o `disable_intra_planar`
- o `disable_intra_mpm_prediction`
- o `disable_intra_chroma_predictor`
- o `disable_inter_tmvp_merge`
- o `disable_inter_merge_candidates`
- o `disable_inter_merge_mode`
- o `disable_inter_chroma_subpel`
- o `disable_inter_bipred`
- o `disable_transform_residual_greater_than_flags`
- o `disable_transform_last_position`
- o `disable_transform_cbf`
- o `disable_cabac_ctx_update`
- o `disable_cabac_split_flag_ctx`
- o `disable_cabac_coeff_sig_ctx`
- o `disable_cabac_coeff_greater1_ctx`
- o `disable_deblock_weak_filter`
- o `disable_deblock_chroma_filter`
- o `disable_deblock_initial_sample_decision`
- o `disable_deblockDepending_on_qp`
- o `disable_high_level_default_checksum_method`
- o `disable_ext_transform_size_64`

- o `disable_ext2_intra_chroma_from_luma`
- o `disable_ext2_transform_select`
- o `disable_ext2_cabac_alt_residual_ctx`

The following 51 restriction flags are set to 0 in the default configuration of the full xvc codec, but are required to be set to 1 in the royalty-free profile of xvc (i.e. the corresponding tools are excluded from the royalty-free profile).

- o `disable_intra_ref_sample_filter`
- o `disable_intra_dc_post_filter`
- o `disable_intra_ver_hor_post_filter`
- o `disable_inter_mvp`
- o `disable_inter_scaling_mvp`
- o `disable_inter_tmvp_mvp`
- o `disable_inter_tmvp_ref_list_derivation`
- o `disable_inter_merge_bipred`
- o `disable_inter_skip_mode`
- o `disable_inter_mvd_greater_than_flags`
- o `disable_transform_adaptive_scan_order`
- o `disable_transform_residual_greater2`
- o `disable_transform_root_cbf`
- o `disable_transform_subblock_csbfb`
- o `disable_transform_sign_hiding`
- o `disable_transform_adaptive_exp_golomb`
- o `disable_cabac_skip_flag_ctx`
- o `disable_cabac_inter_dir_ctx`
- o `disable_cabac_subblock_csbfb_ctx`

- o `disable_cabac_coeff_greater2_ctx`
- o `disable_cabac_coeff_last_pos_ctx`
- o `disable_cabac_init_per_pic_type`
- o `disable_cabac_init_per_qp`
- o `disable_deblock_strong_filter`
- o `disable_deblock_boundary_strength_zero`
- o `disable_deblock_boundary_strength_one`
- o `disable_deblock_weak_sample_decision`
- o `disable_deblock_two_samples_weak_filter`
- o `disable_ext_implicit_last_ctu`
- o `disable_ext_tmvp_full_resolution`
- o `disable_ext_tmvp_exclude_intra_from_ref_list`
- o `disable_ext_ref_list_l0_trim`
- o `disable_ext_implicit_partition_type`
- o `disable_ext_cabac_alt_split_flag_ctx`
- o `disable_ext_cabac_alt_inter_dir_ctx`
- o `disable_ext_cabac_alt_last_pos_ctx`
- o `disable_ext_two_cu_trees`
- o `disable_ext_intra_unrestricted_predictor`
- o `disable_ext_deblock_subblock_size_4`
- o `disable_ext2_intra_67_modes`
- o `disable_ext2_intra_6_predictors`
- o `disable_ext2_inter_adaptive_fullpel_mv`
- o `disable_ext2_inter_affine`

- o `disable_ext2_inter_affine_merge`
- o `disable_ext2_inter_affine_mvp`
- o `disable_ext2_inter_bipred_ll_mvd_zero`
- o `disable_ext2_inter_high_precision_mv`
- o `disable_ext2_inter_local_illumination_comp`
- o `disable_ext2_transform_skip`
- o `disable_ext2_transform_high_precision`
- o `disable_ext2_transform_dst`

2.9. Version handling

The xvc codec has a simple but powerful scheme for version handling. All xvc bitstreams include, in the beginning of each segment header, one syntax element that represents the major version and one syntax element that represents the minor version.

The major version is increased when non-backwards compatible changes are introduced, typically when new technology is added to xvc. When a new major version of xvc is released, all existing decoders need to be updated in order to support bitstreams of the new major version. Existing bitstreams do not have to be updated when the major version of xvc is increased, since new technology is activated only for new bitstreams that uses the new major version.

The minor version is increased when backwards compatible changes are introduced. In practice, this occurs when technology is being disabled through setting their restriction flag equal to one. When a new minor version of xvc is released, existing decoders do not need to be updated immediately, since they already have support for decoding bitstreams in which the disabled technology is turned off. However, the xvc license requires that they are updated at some point so that they do not include support for the technology that is no longer available in xvc. Existing bitstreams have to be updated when the minor version of xvc is increased since the new version of the reference decoder will reject bitstreams with too low minor version (since they make use of technology that is no longer available in xvc). In practice, it is expected to be extremely rare that the minor version has to be increased, but the framework for how to handle it is in place to ensure that there is always a deployable and licensable version of xvc available, and that no patent holder (or

"patent troll") would be able to block the codec from being used altogether.

The separation of major version and minor version makes it possible to always allow for a transition period when changes are introduced to the xvc codec. In the case of a major version increase, new bitstreams will only be created once decoders have been updated to support the new version. In the case of a minor version increase, new decoders will not replace old decoders until existing bitstreams have been updated to use the new minor version. By applying this scheme, the codec can evolve over time without causing any interoperability problems.

2.10. Temporal scalability and parallel decoding

The default coding structure of the xvc codec is a static hierarchical B-picture structure with 15 B-pictures. The length of the hierarchical structure, called SubGOP, is indicated in the Segment Header. A SubGOP length of 16 corresponds to 15 B-pictures in a dyadic hierarchy, forming four temporal layers above temporal layer zero.

Pictures in temporal layers above 0, only predict from temporal layer 0 or pictures with lower temporal layers within the same SubGOP. This makes it possible to scale of an arbitrary number of temporal layers in any SubGOP without affecting decodability of any pictures outside the same SubGOP. This is a very useful property in a software decoder since it makes it possible to adjust the decoding complexity in response to sudden changes in available processing resources. Instead of freezing the video and build up a delay, the framerate can momentarily be reduced and full framerate can be resumed as soon as enough processing resources are available.

A fixed hierarchical coding structure is also very useful in order to enable efficient parallel decoding. The xvc reference decoder includes support for picture level threaded decoding. When decoding bitstreams with SubGOP length 16 a speed-up factor of between 3x and 4x is typically achieved on a CPU with 4 active physical cores.

Providing support for temporal scalability and picture level parallel decoding generally comes at a very low cost in terms of compression performance. However, on very specific sequences there might be a compression penalty due to the constraint of not predicting from previously coded pictures in the same or higher temporal layers.

The picture level parallelism that is currently implemented in xvc can be combined with other tools for parallelism such as tiles,

wavefronts or slices if a higher degree of parallelism is desired, but support for these tools have not yet been added to xvc.

2.11. Parallel encoding

Support for multi-threaded encoding has been added to the xvc software. The implementation uses picture based parallelism and is very similar to the method used in the decoder, where multiple pictures are processed in parallel as soon as they do not depend on each other. This makes it possible to produce exactly the same result for multi-threaded encoding as for single threaded encoding, i.e. there is no loss in compression performance. The multi-threaded encoding scheme increases memory usage linearly with number of threads and due to the prediction structure of reference pictures, it typically takes some time for the utilization to reach its peak performance. As an example, when using 8 threads, typically 128 pictures needs to be encoded before peak utilization is reached.

2.12. Baseline profile

Version 2.0 of xvc includes a royalty-free baseline profile. The baseline profile is a pure subset of the full xvc codec. The baseline profile uses 25 of the 76 coding tools in xvc. Results for the full xvc codec relative to the baseline profile are included below.

3. The xvc reference software

The implementation of the xvc reference software has been made from scratch using C++11. Some of the advantages of C++11 compared to earlier versions of C++ is improved support for type management, improved pointer handling, availability of lambda expressions, threading and useful new keywords such as `auto`. The xvc implementation strictly follows the Google C++ Style Guide [cppStyle] which is a well established formatting recommendation for C++, intended to provide readability and consistency across various C++ project.

The low-complex design of the decoding process of xvc makes it possible to run the decoder efficiently on a large variety of devices. The xvc reference decoder has been demonstrated to run FullHD decoding in realtime on mobile devices, and a JavaScript xvc decoder, based on the reference xvc decoder, is able to run xvc decoding directly in browsers, as shown on the Divideon demo page [xvcDemo]

The total number of lines of code for the second version of the xvc software is around 25,000. This can be compared to the around 200,000 lines of code in the AV1 software.

4. Results

Experimental results for three different test cases are reported; All-Intra, single pass Random-Access and multi-pass Random-Access. The latter two correspond to "High Latency CQP" and "High Latency Unconstrained" from the Video Codec Testing and Quality Measurement I-D [Daedel17]. The results have been generated using the AreWeCompressedYet? framework [AWCY]. The full set of results are available at <https://awcy.divideon.com> [DAWCY].

For xvc [xvcSource], the commit f0b31546a3251aeb43f7928338b12aa349156139, from 2018-02-28 has been used, with command line parameters: -tune 1 -speed-mode 0 -chroma-qp-offset-u -1 -chroma-qp-offset-v -1. The chroma qp offset is used to better match bit distribution between luma and chroma relative to AV1. The quantizer values that have been used for xvc are: 20, 25, 30, 35, 40.

For AV1 [av1Source], the commit 1a7099443894d07fdf3acbb7e12ed04c2d62b9c5, from 2018-02-02 has been used, with build settings: -DCONFIG_EXPERIMENTAL=1 and --tune-content=screen as extra command line argument. The following quantizer values are used: 20, 32, 43, 55 and 63. For multi-pass coding the run "debargha-adopted-0104@2018-01-05T00:55:50.198Z" was copied from AreWeCompressedYet? [AWCY], since that was the most recent run that was finished at the time of this submission. That run was using commit 5a31bc899b308da9b2cacd339d0b19374a74b906 from "2018-01-04"

For HM [hmSource], version 16.17 from 2017-10-18 has been used with the default configuration files provided in the cfg folder of the source code repository. These configuration files correspond to the JCT-VC common test conditions described in JCTVC-Z1100 [hmCtc]. The following quantizer values are used: 20, 25, 30, 35, 40.

For All-Intra, results are presented for the Subset1 test set, and for the two Random-Access cases results are presented for the Objective-1-fast test set [Testset].

The tables below show the average percentual rate difference measured using the Bjontegaard rate difference, also known as BD-rate [Bjontegaard01]. The BD-rate is measured using the following objective metrics: PSNR, PSNR-HVS [Egiazarian2006], SSIM [Wang04] and MSSIM [Wang03].

4.1. All-Intra

In the tables below, results are presented for the Subset1 test set which consist of 50 different still images.

The following table present results of xvc relative to HM with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
Average	-9.3	-33.3	-30.3	-9.8	-9.9	-9.5

All-intra xvc relative to HM

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
Average	-3.1	-7.4	-7.9	-2.1	-2.8	-3.0

All-intra xvc relative to AV1

4.2. Single pass Random-Access

The following table present results of xvc relative to HM with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-16.8	-29.9	-28.9	-15.0	-17.9	-16.8
1080psc	-13.7	-44.5	-40.4	-15.4	-16.7	-17.0
720p	-20.8	-30.0	-32.6	-20.1	-23.8	-22.7
360p	-26.1	-24.7	-28.8	-26.4	-30.2	-29.6
Average	-19.5	-30.7	-31.3	-19.1	-22.0	-21.2

Single pass Random-Access xvc relative to HM

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate). Results are presented for 360p and 720p only, since we were not able to run encodings for the complete test set for AV1 in time.

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
720p	-13.3	-0.8	-4.4	-16.4	-20.2	-20.5
360p	-19.7	-9.6	-3.4	-22.5	-23.0	-26.1
Average	-16.5	-5.2	-3.9	-19.4	-21.6	-23.3

Single pass Random-Access xvc relative to AV1

4.3. Multi-pass Random-Access

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate). The results for AV1 are imported from AreWeCompressedYet? [AWCY].

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-6.0	-4.5	-3.2	-5.6	-10.9	-9.7
1080psc	8.3	18.5	15.8	5.9	6.2	3.9
720p	-0.5	0.4	4.8	-1.4	-6.0	-5.5
360p	-15.9	-6.2	11.2	-19.9	-19.0	-21.2
Average	-5.1	-0.7	4.6	-6.4	-9.4	-9.6

Multi-pass Random-Access xvc relative to AV1

4.4. Full xvc relative to baseline xvc for single pass Random-Access

The following table present results of the full xvc codec relative to the baseline profile of xvc with numbers indicating the percentual bitrate difference for equivalent quality.

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-11.6	-14.6	-14.4	-10.9	-11.9	-11.5
1080psc	-15.1	-23.8	-24.4	-11.5	-12.7	-12.0
720p	-10.2	-14.7	-15.8	-9.9	-11.4	-10.6
360p	-14.9	-18.6	-21.4	-15.0	-16.4	-15.6
Average	-12.5	-16.8	-17.7	-11.7	-12.9	-12.3

Multi-pass Random-Access full xvc relative to baseline xvc

4.5. Full xvc relative to baseline xvc for multi-pass Random-Access

The following table present results of the full xvc codec relative to the baseline profile of xvc with numbers indicating the percentual bitrate difference for equivalent quality.

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-11.5	-14.8	-14.1	-10.6	-11.5	-11.1
1080psc	-15.8	-23.7	-24.7	-12.6	-14.1	-13.1
720p	-10.3	-14.0	-15.1	-10.3	-12.0	-11.1
360p	-14.3	-17.7	-18.5	-14.1	-15.7	-14.7
Average	-12.5	-16.5	-16.8	-11.6	-12.9	-12.2

Single pass Random-Access full xvc relative to baseline xvc

4.6. Computational complexity

The single pass Random-Access configurations have been run on the same platform (Intel Xeon E5-2660). The encoding times and decoding times can be used to estimate the computational complexity when running that particular setting on that particular hardware. On average the encoding time is around 340% higher for AV1 than for xvc. The decoding time is around 70% higher for AV1 than for xvc.

5. Discussion

The xvc codec has been developed with the aim of creating the most efficient video codec that can be made available under a single reasonable license. Different coding tools have been included based solely on their technical merits. Technology is replaced or removed only if the patent holder of that particular technology has requested for it to be removed, or if the patent holder by other means have made it clear that they intend to seek royalties for the technology

outside of the xvc licensing program or initiate patent litigation or a patent lawsuit related to the technology. This approach has made it possible to make use of the best available technology rather than having to compromise, exclude and/or design around technology to create a lesser efficient alternative.

Divideon is in continuous dialog with potential patent holders and has previously sent out a Call for Patents in xvc [xvcCall], to make sure that the license actually covers all the necessary technology to use, implement and distribute conforming xvc implementations. As with all modern video codecs it is not possible to give a definite answer to the patent situation, but what is unique with xvc is the well-defined framework for handling third party patent assertions.

In the Call for Patents in xvc that was sent out by Divideon there was also a request for feedback regarding the idea of defining a royalty-free profile of xvc. The second version of xvc includes such a royalty-free baseline profile as described above.

The xvc codec is brought to the NETVC Working Group as a candidate for the Internet Video Codec. The objectives of the Working Group states that "This WG is chartered to produce a high-quality video codec that meets the following conditions:

1. Is competitive (in the sense of having comparable or better performance) with current video codecs in widespread use.
2. Is optimized for use in interactive web applications.
3. Is viewed as having IPR licensing terms that allow it to be widely implemented and deployed."

The authors of this document believe that xvc is well positioned to fulfill all of these conditions. A framework with a well-defined single license and a royalty-free baseline profile ensures that condition 1 and 3 can both be fulfilled, and the reference software of xvc constitutes a good example of that the codec (and especially the decoder) can be run in realtime on common hardware including mobile devices, and thereby fulfill the second condition.

6. IANA Considerations

This document has no IANA considerations.

7. Security Considerations

This document has no security considerations.

8. Informative References

- [av1Source] Alliance for Open Media, "The av1 source code", n.d., <<https://aomedia.googlesource.com/aom/>>.
- [AWCY] Xiph.Org Foundation, "Are We Compressed Yet?", n.d., <<https://arewecompressedyet.com>>.
- [Bjontegaard01] Bjontegaard, G., "Calculation of average PSNR differences between RD-curves", 2001, <http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc>.
- [cppStyle] Google, "Google C++ Style Guide", n.d., <<https://google.github.io/styleguide/cppguide.html>>.
- [Daede17] Daede, T., Norkin, A., and I. Brailovsky, "Video Codec Testing and Quality Measurement", IETF NETVC Internet-Draft draft-ietf-netvc-testing-06, October 2017.
- [DAWCY] Xiph.Org Foundation, "Are We Compressed Yet? Divideon clone", n.d., <<https://awcy.divideon.com>>.
- [Egiazarian2006] Egiazarian, K., Astola, J., Ponomarenko, N., Lukin, V., Battisti, F., and M. Carli, "Two new full-reference quality metrics based on HVS", Proceedings of the Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics VPQM, January 2006.
- [HEVC] ISO/IEC/ITU-T, "High efficiency video coding", n.d., <<https://www.itu.int/rec/T-REC-H.265>>.
- [hmCtc] Sharman, K. and K. Suehring, "Common test conditions", 2017, <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>.
- [hmSource] ISO/IEC/ITU-T, "The HM source code", n.d., <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>.

- [Testset] Daede, T., "Test Sets", n.d.,
<<https://people.xiph.org/~tdaede/sets/>>.
- [Wang03] Wang, Z., Simoncelli, E., and A. Bovik, "Multiscale structural similarity for image quality assessment", The 37th Asilomar Conference on Signals, Systems Computers Volume 2, November 2003.
- [Wang04] Wang, Z., Bovik, A., Sheikh, H., and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", issn 1057-7149, IEEE transactions on image processing Volume 13, number 4, April 2004.
- [Wenger16] Wenger, S., "On profiles and per-feature Flags", 2016,
<http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=10492>.
- [xvcCall] Divideon, "Call for Patents in xvc", 2018,
<<http://www.releasewire.com/press-releases/call-for-patents-in-xvc-929388.htm>>.
- [xvcDemo] Divideon, "Mobile video streaming with xvc", 2018,
<<https://www.divideon.com/products-and-services/mobile-video-streaming-with-xvc/>>.
- [xvcio] Divideon, "The xvc web page", n.d., <<https://xvc.io/>>.
- [xvcLicense] Divideon, "The xvc license", n.d.,
<<https://xvc.io/license>>.
- [xvcSource] Divideon, "The xvc source code", n.d.,
<<https://github.com/divideon/xvc/tree/dev>>.
- [xvcWp] Samuelsson, J. and P. Hermansson, "Introducing xvc - a Divideon white paper", 2017, <<https://www.divideon.com/wp-content/uploads/2017/09/Introducing-xvc-a-Divideon-whitepaper-v1.0.pdf>>.

Authors' Addresses

Jonatan Samuelsson
Divideon
Kulstoetarvaegen 14
Stockholm 12240
Sweden

Email: jonatan.samuelsson@divideon.com

Per Hermansson
Divideon
Kulstoetarvaegen 14
Stockholm 12240
Sweden

Email: per.hermansson@divideon.com