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 contrained 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

Table of Contents

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:

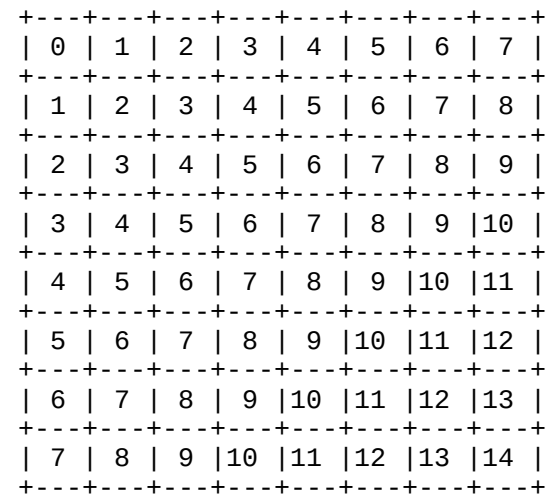

Figure 1: Line number k for pixels following direction 0

Figure 2: Line number k for pixels following direction 1

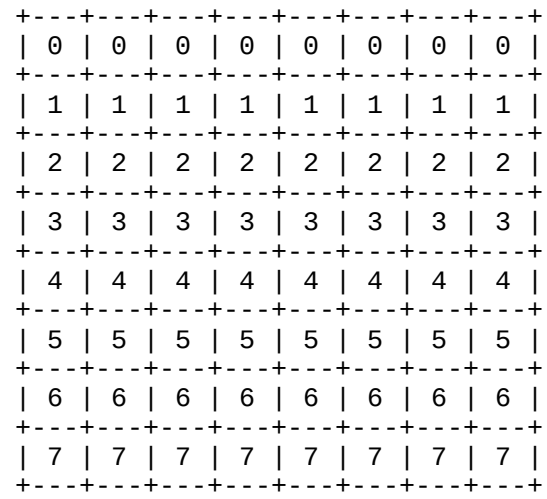

Figure 3: Line number k for pixels following direction 2

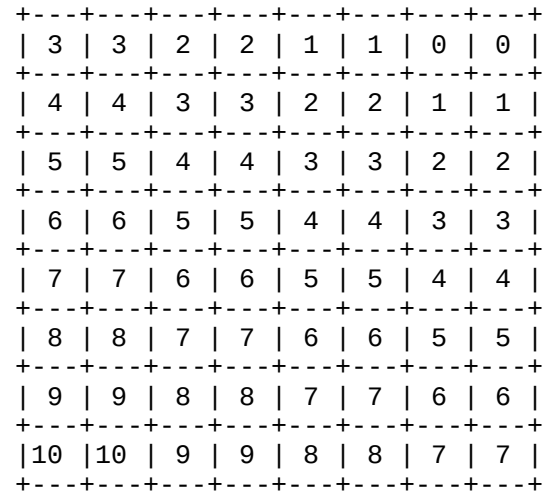

Figure 4: Line number k for pixels following direction 3

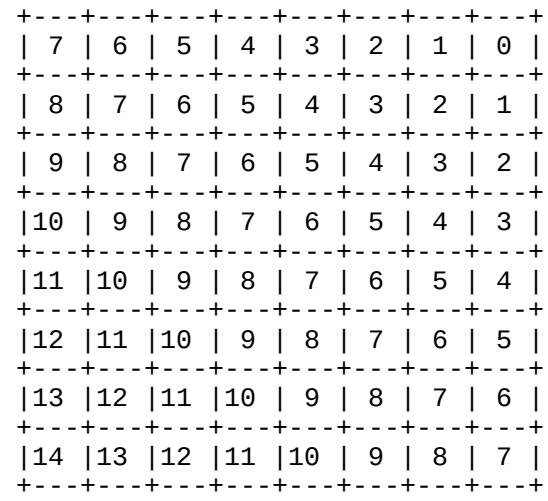

Figure 5: Line number k for pixels following direction 4

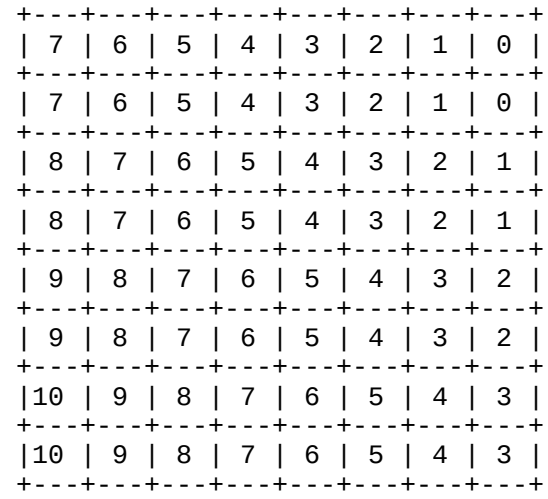

Figure 6: Line number k for pixels following direction 5

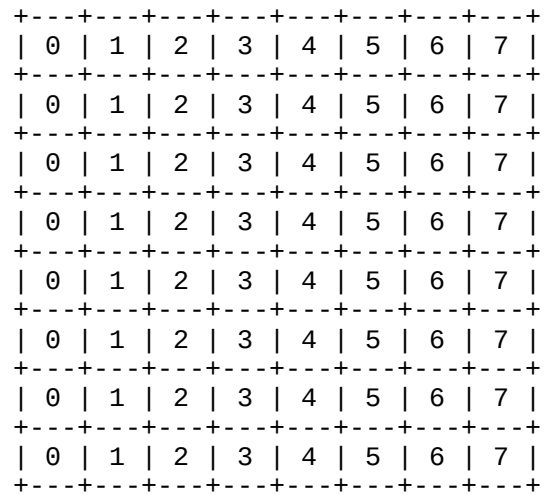

Figure 7: Line number k for pixels following direction 6

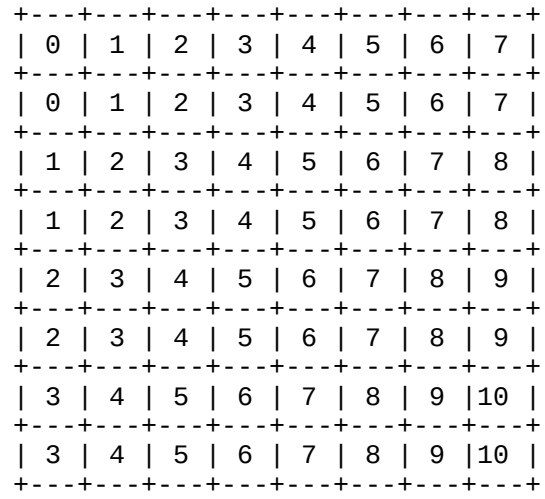

Figure 8: Line number k for pixels following direction 7

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] <- partial[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] + partial[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 algoritm

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) = round(x(i,j) + g(/__ w'_m,n * f(x(i,j)-x(m,m), S', D) +
                                 __
                                 \
                                 /__ 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:

```
            | min(d, max(0, S - floor(d/(2^D - floor(log2(S)))))), d >= 0
f(d,S,D) = <
            | max(d, min(0, floor(d/(2^D - floor(log2(S)))) - S)), d < 0
```

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

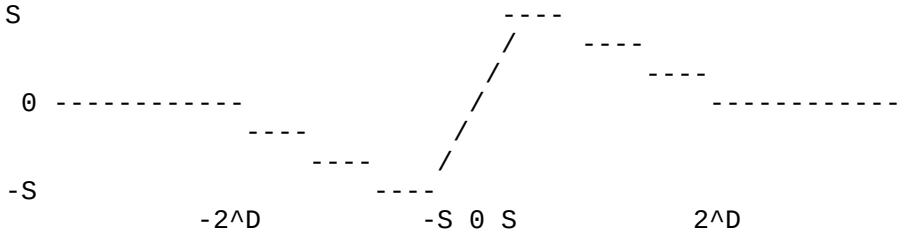

Figure 12: Graph 1

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) = 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:

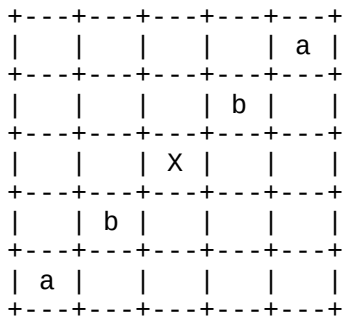

Figure 15: Primary taps (w') for direction 0

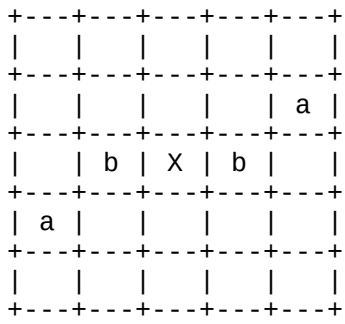

Figure 16: Primary taps (w') for direction 1

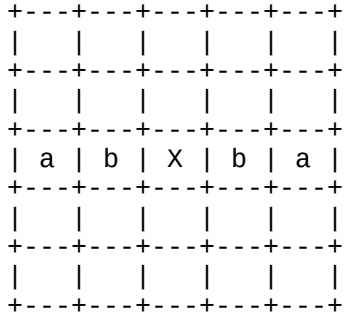

Figure 17: Primary taps (w') for direction 2

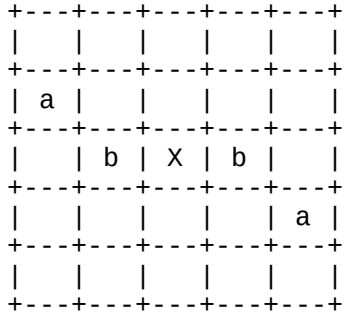

Figure 18: Primary taps (w') for direction 3

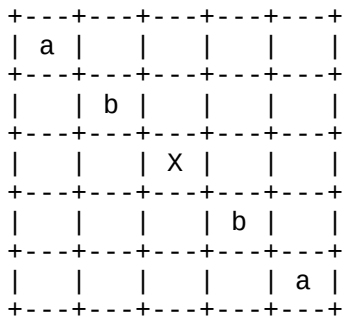

Figure 19: Primary taps (w') for direction 4

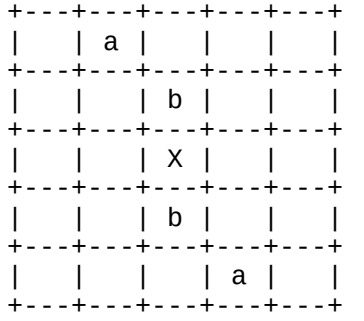

Figure 20: Primary taps (w') for direction 5

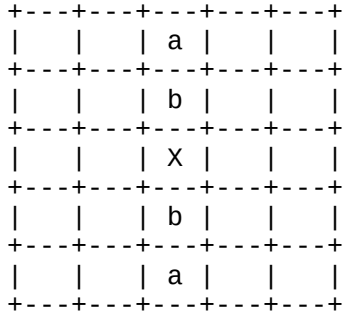

Figure 21: Primary taps (w') for direction 6

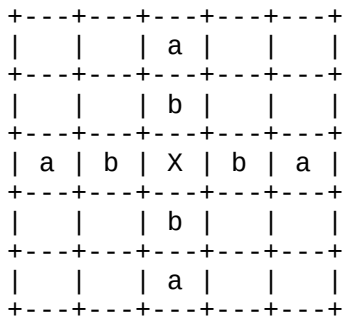

Figure 22: Primary taps (w') for direction 7

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:

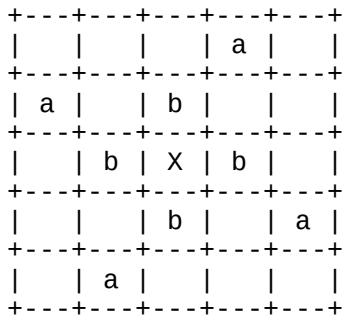

Figure 23: Secondary taps (w'') for direction 0 and 4

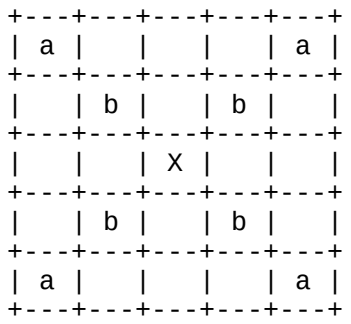

Figure 24: Secondary taps (w'') for direction 1 and 5

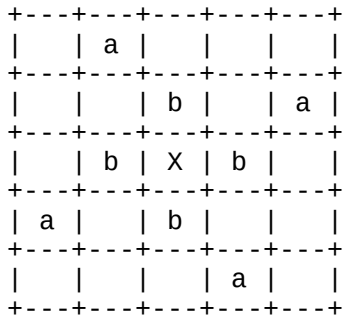

Figure 25: Secondary taps (w'') for direction 2 and 6

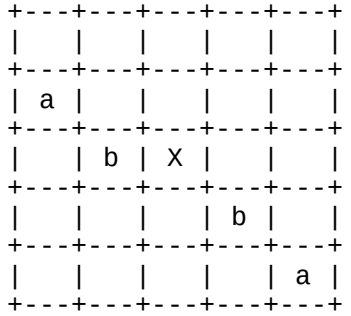

Figure 26: Secondary taps (w'') for direction 3 and 7

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 log2(S) to ensure that the shift value used to compute 2^(D-floor(log2(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 floor(log2(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 optimial 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 particularily 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 effiency, 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 effiency, 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 compexity, 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 compexity, 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