Thor update

High Efficiency, Moderate Complexity Video Codec using only RF IPR

draft-fuldseth-netvc-thor-02
draft-midtskogen-netvc-clpf-02
draft-davies-netvc-qmtx-00

Steinar Midtskogen (Cisco)
IETF 95 – Buenos Aires, AR – April 2016
Thor, a simple and efficient codec

- Designed to be simple, efficient and pragmatic
- Simple both in terms of computation and description
- Uses techniques known to work & improves on those
- Many similarities with H.26x
- Royalty free IPR
  - NOTE WELL: https://datatracker.ietf.org/ipr/2636/
Topics for this update

• Changes since IETF94/November 2015
  – Support for 128x128 superbblocks
  – New constrained low pass filter
  – Weighted quantisation matrices
  – Rate control support
  – Misc tuning

• Updated compression performance
128x128 superblocks

- The encoder can specify whether the SB size is 64x64 (as before) or 128x128.
- 128x128 inverse transform defined as a 32x32 transform and resulting pixels are duplicated (1->4x4)
- Bandwidth reductions: LD: 2.0%, HD: 2.6%
  - Useful for low bitrates and VC (4-5% reductions), mostly < 1% reductions for other material
- About half of the gain comes from 128x128 skip, so the added complexity for the encoder can be kept low
Old constrained low-pass filter

• Previous filter:

\[ X' = X + ((A>X)+(B>X)+(C>X)+(D>X) > 2) - ((A<X)+(B<X)+(C<X)+(D<X) > 2) \]

• Increase/decrease by one if at least three of the four neighbours are larger/smaller.

• Very simple, yet effective if applied to the right blocks.

• Weakness: Only $\pm 1$ adjustments. A conservative filter.
New constrained low-pass filter

A general way to modify a pixel \( x(i, j) \):

\[
y(i, j) = \text{round}(x(i, j) + g(\sum_{m,n \in R} a(m,n) f(x(i, j) - b(m,n)x(m,n)))}
\]

- \( R \) is the region of interest (e.g. a 3x3 neighbourhood)
- \( a(m,n) \) and \( b(m,n) \) are real-valued coefficients
- \( f() \) and \( g() \) are functions, possibly non-linear
- \( \text{round}(x) \) maps \( x \) to an integer in the desired range (e.g. to the nearest integer in 0-255 for 8 bit input)
New constrained low-pass filter

\[ y_{i, j} = \text{round}(x_{i, j}) + g\left( \sum_{m, n \in R} a(m, n) f(x_{i, j} - b(m, n) x(m, n)) \right) \]

- **Special case 1** (almost identical to previous CLPF):
  - \( R = \{x(i-1,j), x(i,-1), x(i+1,j), x(i,j+1)\} \)
  - \( a(m, n) = 0.25 \)
  - \( b(m, n) = 1 \)
  - \( f(x) = \text{clip}(x, -1, 1) \)
  - \( g(x) = x \)
  - \( \text{round}(x) \) maps \( x \) to the nearest integer

- **Special case 2** (traditional FIR filter)
  - \( b(m, n) = 0 \)
  - \( f(x) = g(x) = x \)
New constrained low-pass filter

\[
y(i, j) = \text{round} \left( x(i, j) + g \left( \sum_{m,n \in R} a(m,n) f(x(i,j) - b(m,n)x(m,n)) \right) \right)
\]

- New CLPF:
  - 6 pixels in a cross shape used as input, horizontally oriented to minimise line buffer requirements.
  - \( g(x) = x \), \( \text{round}(x) \) maps to the nearest integer, and \( s \) is filter strength

\[
y(i, j) = \text{round}(x(i,j)) + \frac{1}{4}\text{clip}(x(i-1,j) - x(i,j), -s, s) + \frac{1}{16}\text{clip}(x(i-2,j) - x(i,j), -s, s) + \frac{3}{16}\text{clip}(x(i-1,j) - x(i,j), -s, s) + \frac{3}{16}\text{clip}(x(i+1,j) - x(i,j), -s, s) + \frac{1}{16}\text{clip}(x(i+2,j) - x(i,j), -s, s)
\]

\[
Y = X + \left( 4\text{clip}(A-X,-s,s) + \text{clip}(B-X,-s,s) + 3\text{clip}(C-X,-s,s) + 3\text{clip}(D-X,-s,s) + \text{clip}(E-X,-s,s) + 4\text{clip}(F-X,-s,s) \right) / 16
\]
New constrained low-pass filter

- Slightly more computationally complex:

```c
#define clip(n,l,h) ((h) < ((n)>(l) ? (n) : (l)) ? (h) : ((n)>(l) ? (n) : (l)))

int clpf_pixel(int X, int A, int B, int C, int D, int E, int F, int s)
{
    int delta = 4*clip(A - X, -s, s) + clip(B - X, -s, s) + 3*clip(C - X, -s, s) +
                3*clip(D - X, -s, s) + clip(E - X, -s, s) + 4*clip(F - X, -s, s);
    return X + ((8 + delta - (delta < 0)) >> 4);  // Assumes arithmetic shift
}
```

compared to the old CLPF:

```c
int clpf_pixel(int X, int A, int B, int C, int D)
{
    return X + ((A>X)+(B>X)+(C>X)+(D>X) > 2) - ((A<X)+(B<X)+(C<X)+(D<X) > 2);
}
```

- But still SIMD friendly and simple:
  - 8-bit only arithmetics possible for 8 bit content (offset + saturating subtraction for 9 bit difference)
  - 4.9 instructions per pixel to filter an 8x8 block on ARM/NEON (armv7) using C with intrinsics (gcc 4.8.4).
New constrained low-pass filter

- One of four different strengths signalled at frame level:
  - 0 (off), 1, 2 or 4. Same strength for all filtered blocks in a frame
- Coding blocks encoded as skip (Inter0) not filtered
- The filter can be turned off for individual blocks. The block size is 32, 64 or 128 selected at frame level
- 13 different filter options for the encoder:
  - Off, filter frame with s=1, 2 or 4, filter blocks with s=1, 2 or 4 for block sizes 32x32, 64x64 or 128x128
- RDO is fast but s=2 and QP dependent block sizes work well
- Details:
  
  https://tools.ietf.org/id/draft-midtskogen-netvc-clpf-02.txt
Strength = 2
Strength = 4
Strength = 0
Strength = 8
New constrained low-pass filter

PSNR results, uni-prediction only, low delay, medium complexity:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>BDR</th>
<th>BDR (low br)</th>
<th>BDR (high br)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kimono</td>
<td>-2.7%</td>
<td>-2.3%</td>
<td>-3.4%</td>
</tr>
<tr>
<td>BasketballDrive</td>
<td>-3.3%</td>
<td>-2.5%</td>
<td>-4.5%</td>
</tr>
<tr>
<td>BQTerrace</td>
<td>-7.2%</td>
<td>-4.9%</td>
<td>-9.1%</td>
</tr>
<tr>
<td>FourPeople</td>
<td>-5.7%</td>
<td>-3.9%</td>
<td>-8.6%</td>
</tr>
<tr>
<td>Johnny</td>
<td>-5.9%</td>
<td>-4.0%</td>
<td>-9.0%</td>
</tr>
<tr>
<td>ChangeSeats</td>
<td>-6.4%</td>
<td>-3.4%</td>
<td>-10.8%</td>
</tr>
<tr>
<td>HeadAndShoulder</td>
<td>-8.6%</td>
<td>-2.6%</td>
<td>-18.8%</td>
</tr>
<tr>
<td>TelePresence</td>
<td>-5.9%</td>
<td>-3.1%</td>
<td>-10.7%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-5.7%</strong></td>
<td><strong>-3.3%</strong></td>
<td><strong>-9.4%</strong></td>
</tr>
<tr>
<td>Previous average</td>
<td>-4.2%</td>
<td>-1.7%</td>
<td>-7.4%</td>
</tr>
</tbody>
</table>
New constrained low-pass filter

PSNR results, uni-/bi-prediction, low delay, medium complexity:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>BDR</th>
<th>BDR (low br)</th>
<th>BDR (high br)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kimono</td>
<td>-2.2%</td>
<td>-1.8%</td>
<td>-2.7%</td>
</tr>
<tr>
<td>BasketballDrive</td>
<td>-2.6%</td>
<td>-2.5%</td>
<td>-2.7%</td>
</tr>
<tr>
<td>BQTerrace</td>
<td>-4.1%</td>
<td>-3.1%</td>
<td>-4.7%</td>
</tr>
<tr>
<td>FourPeople</td>
<td>-4.0%</td>
<td>-2.9%</td>
<td>-5.3%</td>
</tr>
<tr>
<td>Johnny</td>
<td>-3.5%</td>
<td>-2.7%</td>
<td>-4.6%</td>
</tr>
<tr>
<td>ChangeSeats</td>
<td>-4.2%</td>
<td>-3.0%</td>
<td>-6.1%</td>
</tr>
<tr>
<td>HeadAndShoulder</td>
<td>-4.1%</td>
<td>-2.9%</td>
<td>-6.1%</td>
</tr>
<tr>
<td>TelePresence</td>
<td>-2.8%</td>
<td>-1.9%</td>
<td>-4.3%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-3.4%</strong></td>
<td><strong>-2.6%</strong></td>
<td><strong>-4.6%</strong></td>
</tr>
<tr>
<td>Previous average</td>
<td>-1.8%</td>
<td>-0.9%</td>
<td>-3.1%</td>
</tr>
</tbody>
</table>
Weighted quantisation matrices

The human contrast sensitivity varies with frequency:
Weighted quantisation matrices

- Different scaling factors can be applied to each coefficient during dequantisation.
- The aim is to use the contrast sensitivity function to reduce the quantisation errors in some frequencies and increase them in less important frequencies.
- One matrix for every combination of transform size (6), inter/intra (2), video component (3), and quantisation level (12) – 432 matrices in all (derivable from a smaller set).
- The range of QP-dependent matrices and ability to signal strength reduce the need for custom matrices.
- Details:
  
  https://tools.ietf.org/id/draft-davies-netvc-thor-qmtx-00.txt
# Weighted quantisation matrices

**BDR results for weighted matrices:**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Fast-SSIM high delay</th>
<th>Fast-SSIM low delay</th>
<th>PSNR high delay</th>
<th>PSNR low delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kimono</td>
<td>-5.4%</td>
<td>-0.5%</td>
<td>2.7%</td>
<td>1.9%</td>
</tr>
<tr>
<td>BasketballDrive</td>
<td>-6.7%</td>
<td>-4.4%</td>
<td>2.5%</td>
<td>0.8%</td>
</tr>
<tr>
<td>BQTerrace</td>
<td>-20.0%</td>
<td>-15.2%</td>
<td>1.8%</td>
<td>0.9%</td>
</tr>
<tr>
<td>FourPeople</td>
<td>-12.8%</td>
<td>-12.7%</td>
<td>-0.5%</td>
<td>-1.4%</td>
</tr>
<tr>
<td>Johnny</td>
<td>-18.9%</td>
<td>-18.9%</td>
<td>-1.2%</td>
<td>-1.3%</td>
</tr>
<tr>
<td>ChangeSeats</td>
<td>-15.4%</td>
<td>-10.8%</td>
<td>2.3%</td>
<td>1.0%</td>
</tr>
<tr>
<td>HeadAndShoulder</td>
<td>-9.1%</td>
<td>-13.4%</td>
<td>-0.6%</td>
<td>-1.9%</td>
</tr>
<tr>
<td>TelePresence</td>
<td>-27.9%</td>
<td>-21.7%</td>
<td>3.9%</td>
<td>1.3%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-14.5%</strong></td>
<td><strong>-12.2%</strong></td>
<td><strong>1.4%</strong></td>
<td><strong>0.2%</strong></td>
</tr>
</tbody>
</table>
Rate control

• Constant quality or constant bitrate now configurable
• Delta-QP can be transmitted for every non-skip superblock
• Sliding window operation
• Designed to react quickly (window size = one frame), suitable for videoconferencing
Misc tuning

• Misc tuning giving small gains or simplifications:
  – Interpolation filter coefficients reduced from 7 to 6 bits
  – Reduced depth at which interpolated references are used
  – Improved VLC for motion vectors
  – etc

• Source code:
  – Available at: github.com/cisco/thor
Performance, low delay

• Anchor:
  – HM13.0 (HEVC reference software)
  – Low-delay B configuration

• Thor:
  – Same constraints as the anchor

• VP9: `--cpu-used=0 --end-usage=q --cq-level=$q --kf-min-dist=999 --frame-parallel=0 --tile-columns=0 --threads=1 --ivf -p 1 --auto-alt-ref=0 --lag-in-frames=0`

• x265: `--I -1 --no-wpp --tune psnr -p veryslow --qp $q -bframes 0 --qpf $q.txt`

• Complexity: FourPeople at QP 32 on a single core

Note: HM, Thor and x265 have fixed QP variation, VP9 adapts dynamically.
Frame rate vs compression LD

Encoder frame rate vs. BDR - low delay

- VP9
- X265
- H.265 ref
- Thor-IETF94
- Thor-IETF95
## Performance, low delay

<table>
<thead>
<tr>
<th>Class</th>
<th>Sequence</th>
<th>Thor</th>
<th>VP9</th>
<th>x265</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B</td>
<td>Kimono</td>
<td>14.4</td>
<td>20.9</td>
<td>14.1</td>
</tr>
<tr>
<td></td>
<td>ParkScene</td>
<td>18.6</td>
<td>28.9</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>Cactus</td>
<td>15.4</td>
<td>12.1</td>
<td>21.5</td>
</tr>
<tr>
<td></td>
<td>BasketballDrive</td>
<td>24.4</td>
<td>33.0</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>BQ Terrace</td>
<td>26.9</td>
<td>82.9</td>
<td>44.9</td>
</tr>
<tr>
<td>Class E</td>
<td>FourPeople</td>
<td>4.3</td>
<td>7.8</td>
<td>22.5</td>
</tr>
<tr>
<td></td>
<td>Johnny</td>
<td>7.3</td>
<td>34.9</td>
<td>30.8</td>
</tr>
<tr>
<td></td>
<td>KristenAndSara</td>
<td>1.5</td>
<td>9.6</td>
<td>20.3</td>
</tr>
<tr>
<td>Internal</td>
<td>ChangeSeats</td>
<td>13.9</td>
<td>17.4</td>
<td>12.8</td>
</tr>
<tr>
<td></td>
<td>HeadAndShoulder</td>
<td>-5.6</td>
<td>38.0</td>
<td>34.8</td>
</tr>
<tr>
<td></td>
<td>TelePresence</td>
<td>9.5</td>
<td>23.3</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td>WhiteBoard</td>
<td>8.7</td>
<td>35.1</td>
<td>24.3</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>11.4</strong></td>
<td><strong>28.7</strong></td>
<td><strong>22.4</strong></td>
</tr>
</tbody>
</table>
Performance, high delay

- Anchor:
  - HM13.0 (HEVC reference software)
  - Random access without periodic I frames

- Thor:
  - Same constraints as the anchor

- VP9: `--cpu-used=0 --end-usage=q --cq-level=$q --kf-min-dist=999 --frame-parallel=0 --tile-columns=0 --threads=1 --ivf -p 2 -auto-alt-ref=1 -lag-in-frames=255`

- x265: `-I -1 --no-wpp --tune psnr -p veryslow --qp $q`

- Complexity: FourPeople at QP 32 on a single core

Note: HM and Thor have fixed QP variation, x265 and VP9 adapt dynamically. VP9 did a two-pass encode, the others one-pass.
Frame rate vs compression HD

Encoder frame rate vs. BDR - high delay

- VP9
- X265
- H.265 ref
- Thor-IETF94
- Thor-IETF95
## Performance, high delay

<table>
<thead>
<tr>
<th>Class</th>
<th>Sequence</th>
<th>Thor</th>
<th>VP9</th>
<th>x265</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B</td>
<td>Kimono</td>
<td>17.0</td>
<td>14.8</td>
<td>20.3</td>
</tr>
<tr>
<td></td>
<td>ParkScene</td>
<td>18.1</td>
<td>16.7</td>
<td>26.5</td>
</tr>
<tr>
<td></td>
<td>Cactus</td>
<td>14.0</td>
<td>18.9</td>
<td>17.2</td>
</tr>
<tr>
<td></td>
<td>BasketballDrive</td>
<td>26.7</td>
<td>19.2</td>
<td>13.3</td>
</tr>
<tr>
<td></td>
<td>BQTerrace</td>
<td>30.1</td>
<td>24.1</td>
<td>19.7</td>
</tr>
<tr>
<td>Class E</td>
<td>FourPeople</td>
<td>3.1</td>
<td>1.3</td>
<td>26.7</td>
</tr>
<tr>
<td></td>
<td>Johnny</td>
<td>6.1</td>
<td>13.2</td>
<td>28.4</td>
</tr>
<tr>
<td></td>
<td>KristenAndSara</td>
<td>0.0</td>
<td>11.4</td>
<td>23.0</td>
</tr>
<tr>
<td>Internal</td>
<td>ChangeSeats</td>
<td>12.7</td>
<td>14.1</td>
<td>18.3</td>
</tr>
<tr>
<td></td>
<td>HeadAndShoulder</td>
<td>-3.3</td>
<td>-1.9</td>
<td>21.0</td>
</tr>
<tr>
<td></td>
<td>TelePresence</td>
<td>11.8</td>
<td>14.9</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td>WhiteBoard</td>
<td>6.0</td>
<td>4.8</td>
<td>24.9</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>11.9</strong></td>
<td><strong>12.6</strong></td>
<td><strong>21.6</strong></td>
</tr>
</tbody>
</table>