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 superblocks
  - New constrained low pass filter
  - Weighted quantisation matrices
  - Rate control support
  - Misc tuning
- Updated compression performance

128x128 superblocks

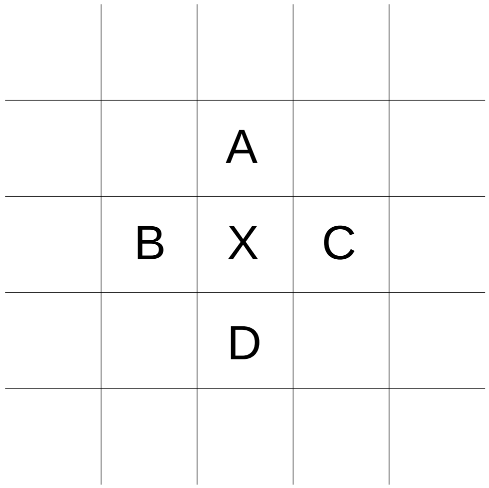
- The encoder can specify whether the SB size is 64x64 (as before) or 128x128.
- 128x128 inverse transform defined as a 32x32 transform and resulting pixels are duplicated (1->4x4)
- Bandwidth reductions: LD: 2.0%, HD: 2.6%
  - Useful for low bitrates and VC (4-5% reductions), mostly < 1% reductions 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 ±1 adjustments. A conservative filter.

New constrained low-pass filter

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

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

- 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
- 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

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

- 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) = clip(x, -1, 1)
  - g(x) = x
  - 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

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

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

*y(i, j) = round(x(i,j) + 1/4*clip(x(i,j-1) – x(i,j), -s, s) +*
*1/16*clip(x(i-2,j) – x(i,j), -s, s) +*
*3/16*clip(x(i-1,j) – x(i,j), -s, s) +*
*3/16*clip(x(i+1,j) – x(i,j), -s, s) +*
*1/16*clip(x(i+2,j) – x(i,j), -s, s) +*
*1/4*clip(x(i,j+1) – x(i,j), -s, s)* or:

		A		
B	C	X	D	E
		F		

Y = X + (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))/16

New constrained low-pass filter

- Slightly more computationally complex:

```
#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:

```
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 = 0

Strength = 4

Strength = 0

Strength = 8

New constrained low-pass filter

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

Sequence	BDR	BDR (low br)	BDR (high br)
Kimono	-2.7%	-2.3%	-3.4%
BasketballDrive	-3.3%	-2.5%	-4.5%
BQTerrace	-7.2%	-4.9%	-9.1%
FourPeople	-5.7%	-3.9%	-8.6%
Johnny	-5.9%	-4.0%	-9.0%
ChangeSeats	-6.4%	-3.4%	-10.8%
HeadAndShoulder	-8.6%	-2.6%	-18.8%
TelePresence	-5.9%	-3.1%	-10.7%
Average	**-5.7%**	**-3.3%**	**-9.4%**
Previous average	-4.2%	-1.7%	-7.4%

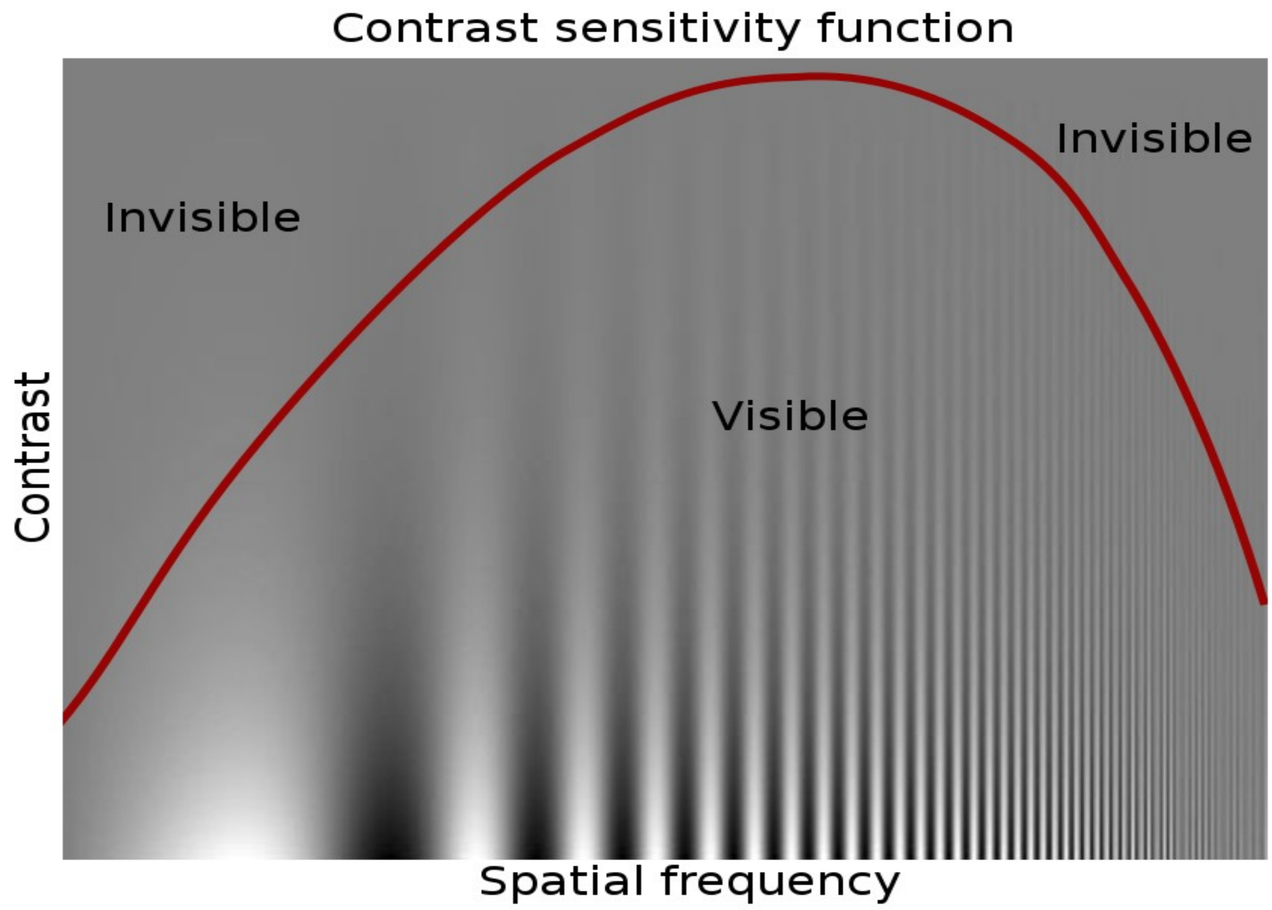
New constrained low-pass filter

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

Sequence	BDR	BDR (low br)	BDR (high br)
Kimono	-2.2%	-1.8%	-2.7%
BasketballDrive	-2.6%	-2.5%	-2.7%
BQTerrace	-4.1%	-3.1%	-4.7%
FourPeople	-4.0%	-2.9%	-5.3%
Johnny	-3.5%	-2.7%	-4.6%
ChangeSeats	-4.2%	-3.0%	-6.1%
HeadAndShoulder	-4.1%	-2.9%	-6.1%
TelePresence	-2.8%	-1.9%	-4.3%
Average	**-3.4%**	**-2.6%**	**-4.6%**
Previous average	-1.8%	-0.9%	-3.1%

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:

Sequence	Fast-SSIM high delay	Fast-SSIM low delay	PSNR high delay	PSNR low delay
Kimono	-5.4%	-0.5%	2.7%	1.9%
BasketballDrive	-6.7%	-4.4%	2.5%	0.8%
BQTerrace	-20.0%	-15.2%	1.8%	0.9%
FourPeople	-12.8%	-12.7%	-0.5%	-1.4%
Johnny	-18.9%	-18.9%	-1.2%	-1.3%
ChangeSeats	-15.4%	-10.8%	2.3%	1.0%
HeadAndShoulder	-9.1%	-13.4%	-0.6%	-1.9%
TelePresence	-27.9%	-21.7%	3.9%	1.3%
Average	**-14.5%**	**-12.2%**	**1.4%**	**0.2%**

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

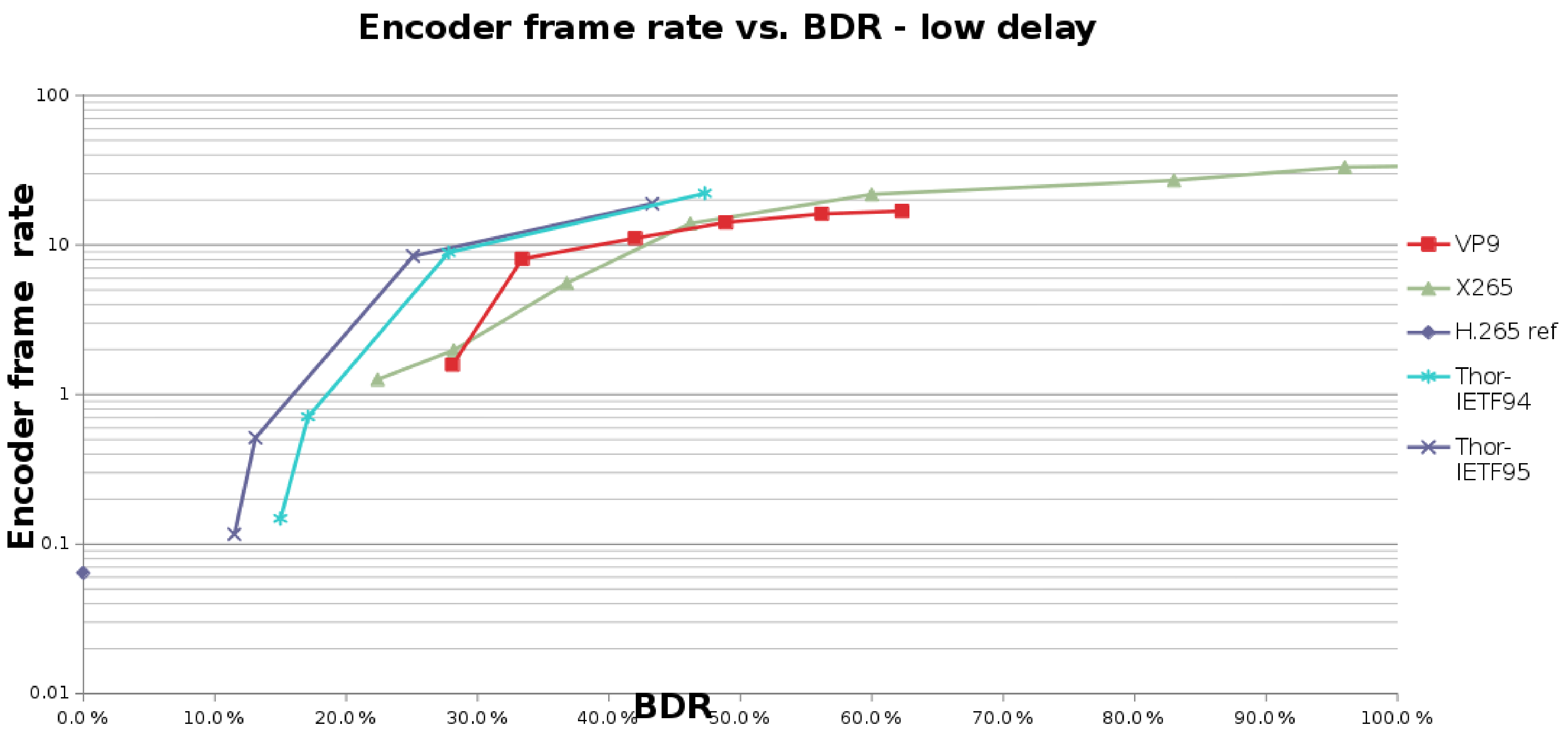
- Misc tuning giving small gains or simplifications:
  - Interpolation filter coefficients reduced from 7 to 6 bits
  - Reduced depth at which interpolated references are used
  - Improved VLC for motion vectors
  - etc
- Source code:
  - Available at: github.com/cisco/thor

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 --qpfile $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



Performance, low delay

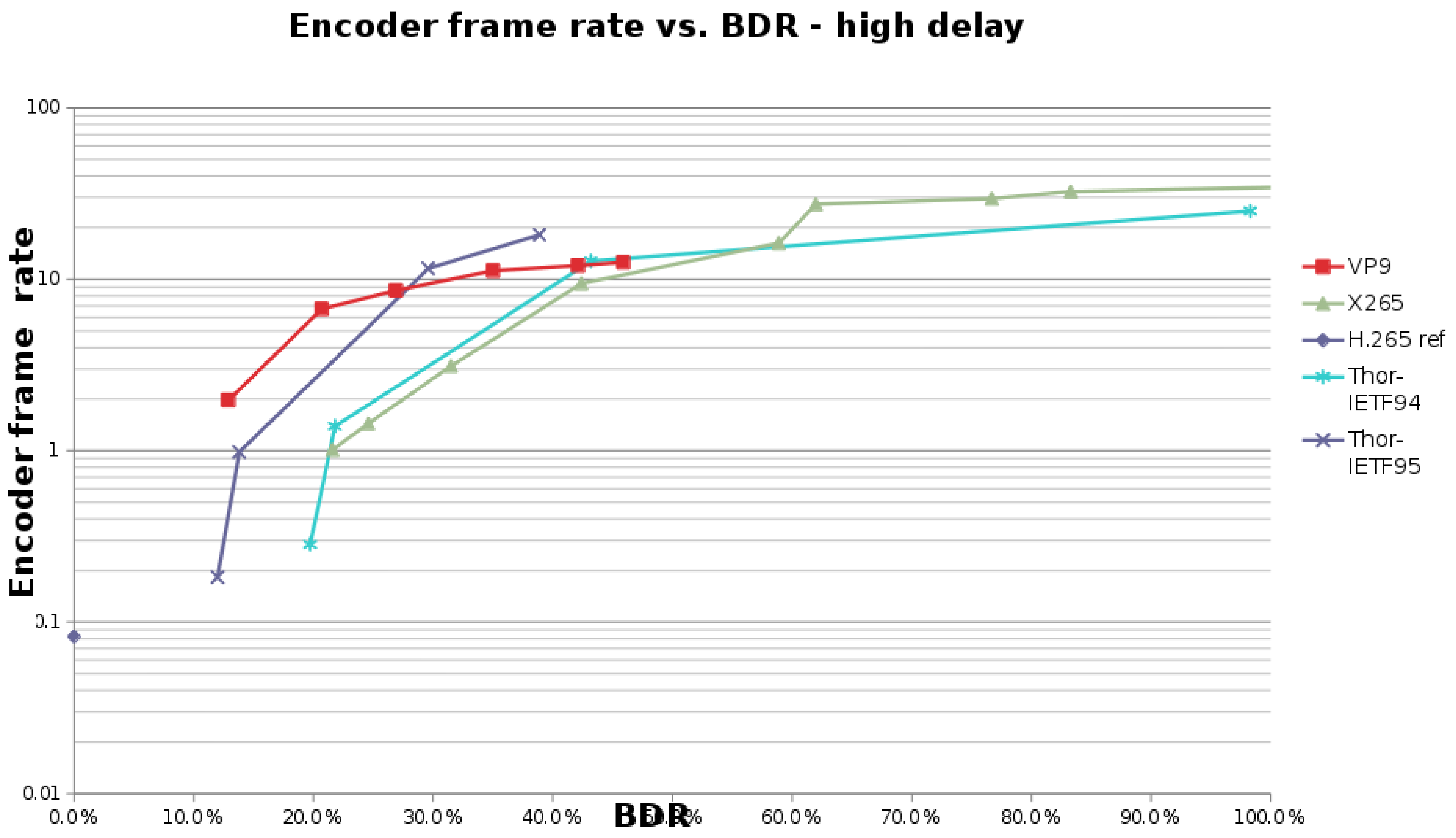
BDR vs. HM13.0 (%)				
Class	**Sequence**	**Thor**	**VP9**	**x265**
Class B	Kimono	14.4	20.9	14.1
	ParkScene	18.6	28.9	16.4
	Cactus	15.4	12.1	21.5
	BasketballDrive	24.4	33.0	14.0
	BQTerrace	26.9	82.9	44.9
Class E	FourPeople	4.3	7.8	22.5
	Johnny	7.3	34.9	30.8
	KristenAndSara	1.5	9.6	20.3
Internal	ChangeSeats	13.9	17.4	12.8
	HeadAndShoulder	-5.6	38.0	34.8
	TelePresence	9.5	23.3	11.9
	WhiteBoard	8.7	35.1	24.3
	Average	**11.4**	**28.7**	**22.4**

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



Performance, high delay

BDR vs. HM13.0 (%)				
Class	**Sequence**	**Thor**	**VP9**	**x265**
Class B	Kimono	17.0	14.8	20.3
	ParkScene	18.1	16.7	26.5
	Cactus	14.0	18.9	17.2
	BasketballDrive	26.7	19.2	13.3
	BQTerrace	30.1	24.1	19.7
Class E	FourPeople	3.1	1.3	26.7
	Johnny	6.1	13.2	28.4
	KristenAndSara	0.0	11.4	23.0
Internal	ChangeSeats	12.7	14.1	18.3
	HeadAndShoulder	-3.3	-1.9	21.0
	TelePresence	11.8	14.9	20.0
	WhiteBoard	6.0	4.8	24.9
	Average	**11.9**	**12.6**	**21.6**