Dynamic optimizer — a perceptual video encoding optimization framework

By Ioannis Katsavounidis, Sr. Research Scientist, Video Algorithms

Netflix Technology Blog Follow
Mar 5, 2018 · 16 min read

Motivation

Video encoding has fueled academic research over the past 25 years and enabled compelling products and services. Many companies are built around video encoding and transmission — Netflix and Google's YouTube are two prime examples of video-centric companies. The fundamentals of video encoding haven't changed for all these years, as modern video streams are produced with the same type of encoding parameters that have been used since MPEG-1 [1]: a certain frame resolution is chosen, together with a group-of-pictures (GOP) structure that imposes periodic Intra pictures, and a target bitrate that is (approximately) met by either a single-pass or two-passes over the input video frames.

Companies have struggled to fine-tune additional parameters in video codecs, creating what is commonly referred to in the industry as a good "recipe". These recipes have been typically created and customized by human inspection of resulting encodes on a selected set of a few titles and have been kept fixed for a very long time.

At the same time, improvements in core video codec tools have led to spectacular reduction in bitrate savings — one can achieve the same quality with an HEVC [2] encoder, while using just a fraction (about 30%) of the bits required by MPEG-1. This improvement, although, which has always been measured using mean-squared-error (MSE), wasn't always accompanied by equally impressive results, when encodes were

evaluated by human observers. The magic number to claim when a new codec was being developed has been “50%”. H264/AVC [3] claimed 50% less bits than MPEG-2 [4] and HEVC claimed 50% less bits than AVC. Yet, in practical systems, these savings never quite materialized — the best estimates on what benefits one sees from an incremental change in video codecs is closer to 40% [5].

Parallel to the standardization efforts at ISO and ITU, Google has been developing their own family of royalty-free video codecs; its latest addition was VP9 [6], first introduced in 2013 and finalized in 2014. VP9 built on the earlier success of VP8 and the line of “True Motion” codecs developed by On2 Technologies, acquired by Google in 2010.

Keeping in mind that most of the video codec improvements carried a very heavy computational overhead on both decoding and — mainly — encoding complexity, one already understands that the newer and more efficient codecs required an ever increasing complexity in order to be deployed in a commercial video transmission service. One typically sees a factor between 5–10 in encoder complexity increase with each generation of video codecs — while the corresponding increase in decoding complexity is typically by a factor of 2.

If one accepts the increased complexity that comes with newer and more efficient codecs, a bigger question is: what can we do at the system level, for example, in the way we connect video frames as input to an encoder or how we use the output of a video decoder to render it on a screen, to further improve the video quality as perceived by human observers who consume all these hours of video today?

The keywords in this new approach, presented here, are the following:

- **Perceptual**: the whole purpose of video encoding is to compress visual information in a way that makes it pleasing to the human eye; Mean-squared-error (MSE), typically used for encoder decisions, is a number that doesn’t always correlate very nicely with human perception.
- **Complexity**: just like we spend an increasing amount of complexity within a video codec, we can afford to spend some outside of it, as well.
- **Look-ahead**: unlike broadcast TV, where one is forced to make encoding decisions on-the-fly or with minimal delay, in video-on-demand services, video sequences

are available in their entirety and can thus be pre-analyzed multiple times to improve quality.

Shot-based encoding

For the remainder of this tech blog, we assume the reader is familiar with the basics of adaptive streaming, such as

- Multiple coded representations of the same visual content in different resolutions and/or qualities, using a basic unit of processing, referred to as “streaming segment”
- Delivery of encoded segments from a server, as requested by a streaming client, that belong to different representations in order to accommodate varying channel conditions (bitstream switching)
- Temporal alignment of segments among different coded representations of the same visual content to allow bitstream switching

Interested readers can refer to a number of available adaptive streaming tutorials, such as this Wiki page [7].

Chunked encoding

In a previous Netflix tech-blog [8], published in Dec. 2015, we described how encoding on the cloud benefits greatly from “chunked” encoding. This translates into breaking a long video sequence, e.g. of 1 hour duration, in multiple chunks, each of a certain duration — for example, 20 chunks, each 3 min. long. We then perform encoding of each chunk independently with a certain encoding recipe, concatenate or “assemble” the encodes and thus obtain an encoded version of the entire video sequence.

Among the advantages of chunked encoding, the most important is that it allows for a robust system to be built on the cloud using software video encoding. If and when cloud instances fail to complete a certain encode, it requires re-processing the corresponding chunk only, instead of restarting an entire hour-long video encode. One can also see the reduction in end-to-end delay, since different chunks can be encoded in parallel; thus achieving almost infinite scalability in the overall encoding system.

There are some penalties that come with chunked encoding — namely the fact that a video encoder operating over the full hour-long sequence, especially in two-pass mode, can preview what is following and therefore do better long-term bitrate allocation; thus achieving better overall quality at the same bitrate. Yet, the advantages that come from chunked encoding outweigh these penalties.

Per-title and per-chunk encode optimization

At Netflix, we have been constantly improving video quality for our members all over the world. One major milestone in our continuous efforts has been "Per-title encode optimization", described in great detail in our techblog, posted in Dec. 2015 [9]. Per-title encode optimization introduced the concept of customizing encoding according to complexity, which translates to proper resolution and bitrate selection for each video sequence we have in our catalog. This provided significant improvement over our previous fixed resolution/bitrate ladder generation, by taking into account the characteristics of video — amount of motion, level of detail, colorfulness — and optimizing coding efficiency by selecting encoding parameters that better fit each title. Another important milestone has been "per-chunk encode optimization", introduced in Dec. 2016 as part of our "Mobile encodes for downloads" initiative, explained in more detail in this Netflix tech blog [10]. The concept of equalizing rate-distortion slopes, discussed in more detail in a subsequent section, was also used in that work and provided significant improvements. In fact, one can consider the current work a natural extension of the "Per-title encode optimization" and "Per-chunk encode optimization"; we can call it "Perceptual per-shot encode optimization".

From chunks to shots

In an ideal world, one would like to chunk a video and impose different sets of parameters to each chunk, in a way to optimize the final assembled video. The first step in achieving this perfect bit allocation is to split video in its natural atoms, consisting of frames that are very similar to each other and thus behave similarly to changes to encoding parameters — these are the "shots" that make up a long video sequence. Shots are portions of video with a relatively short duration, coming from the same camera under fairly constant lighting and environment conditions. It captures the same or similar visual content, for example, the face of an actor standing in front of a tree and — most important — it is uniform in its behavior when changing coding parameters. The natural boundaries of shots are established by relatively simple algorithms, called shot-change detection algorithms, which check the amount of differences between pixels that belong to consecutive frames, as well as other statistics.

When that amount of difference exceeds certain fixed or dynamically adapted threshold, a new shot boundary is announced.

There are cases, such as cross-fades or other visual effects that can be applied on the boundary between two consecutive shots, which can be dealt with by more sophisticated algorithms.

The end result of a shot-change detection algorithm is a list of shots and their timestamps. One can use the resulting shots as the basic encoding block, instead of a fixed-length chunk. That provides for a few really unique opportunities:

1. The placement of Intra frames can now be "consistently irregular", a term that means **(a)** Intra frames can be placed in a "random" place, for example the first 4 Intra frames can be at times 0, 2, 5, 7 secs. and (**b**) Yet, temporal positions are always aligned among encodes of the same title, in order words, the location of the first 4 Intra frames remains at 0, 2, 5, 7 secs. for **all** encodes of this title.
2. The irregular placement of Intra frames results in minimum coding overhead; keep in mind that Intra frames are the least efficient among the 3 different types (I/P/B) used in video coding, and thus one wishes to minimize their presence in an encoded video.
3. Seeking in a long video sequence now leads to natural points of interest, which are signaled by shot boundaries.
4. There is no prediction penalty when encoding shots independently: if one instead places an Intra frame in the middle of a shot, this breaks the shot into parts that, when coded independently instead of a single unit, require more bits, since pixels after the Intra frame can't reference their similar counterparts in frames before the Intra frame.
5. Any significant encoding parameter change between consecutive shots is much less likely to be noticed by the human eye, since the disruption incurred by the different visual content in different shots is far more disruptive to human visual system than any possible encoding parameter — such as resolution/quality — change.
6. Within a homogeneous set of frames, such as those that belong to the same shot, there is much less need to use rate-control, since very simple coding schemes, such as the fixed-quantization parameter ("fixed QP") mode, supported by virtually all existing video encoders, offers a very consistent video quality, with almost minimal

bitrate variation. In fact, “fixed QP” has always been used during development of video codecs, since almost all sequences used for testing in MPEG, ITU and other standards bodies, consist of single-shot video chunks.

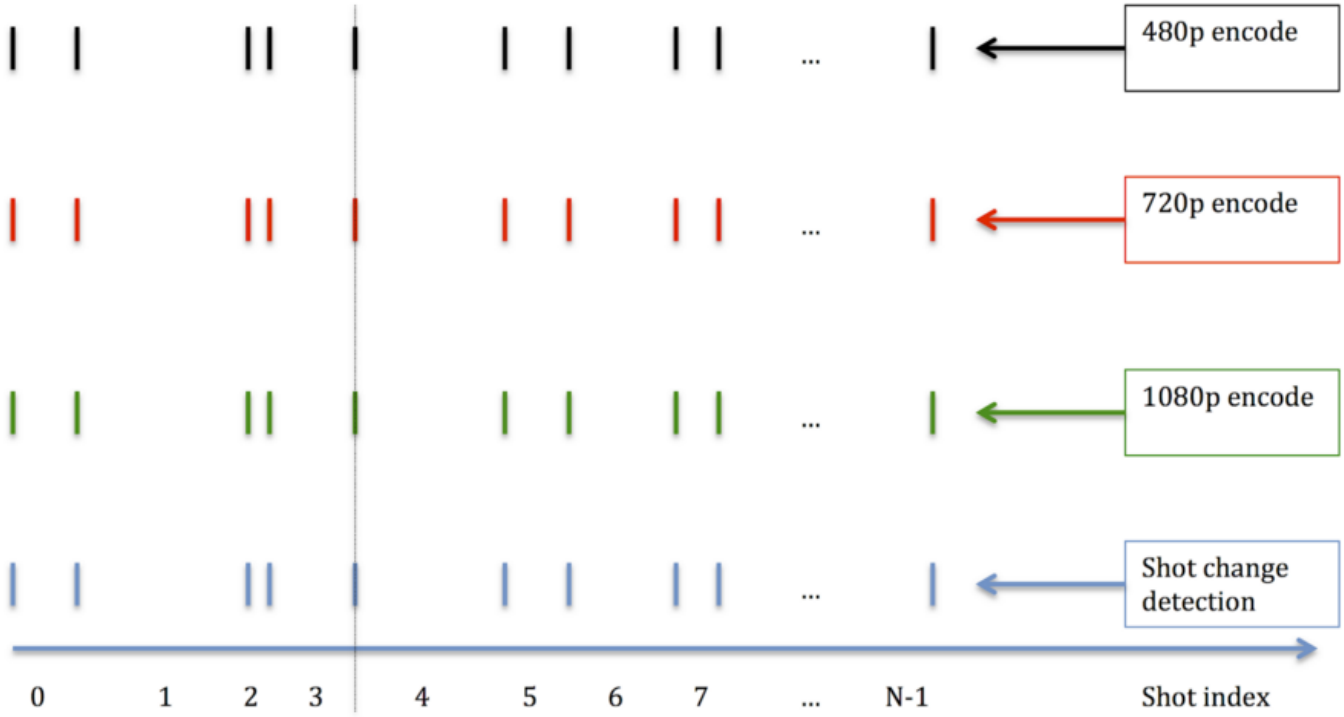


Fig. 1: “Consistently irregular” Intra (key) frame placement on shot boundaries. Key frames are temporally aligned with shot boundaries for all encodes

VMAF as a perceptual video quality metric

In another Netflix tech-blog [11], published in June 2016, we explained the Video Multi-method Assessment Fusion (VMAF) quality metric, developed in-house and then open-sourced for the entire video community to benefit.

Key features of VMAF are the following:

1. It is a full-reference metric, which means it can be applied wherever the original, undistorted version of a video sequence is available, as well as a distorted one.
2. It takes into account both compression and up/down scaling artifacts, by upsampling decoded video frames to a common reference frame size (1920x1080). In this way, one can use VMAF to assess quality of encoded video at different resolutions. In particular, it can be used to compare encoded versions of the same title at different resolutions and help decide which one is better.

3. It relies on existing image quality metrics (VIF, DLM), properly modified to cover multiple scales of resolution, as well as the amount of motion between consecutive video frames in a video sequence as features that are input in a machine-learned set of weights. The final score is the result of combining these elementary features in a support vector machine (SVM) regressor.

4. Calibration and training of the weights used in VMAF has been performed by collecting subjective data from actual observers who provided the ground-truth data that VMAF was then fit against. The content used to train VMAF is a representative subset of the Netflix catalog, therefore it is understood that its performance has been tuned to our use-case. Yet, the VMAF framework is general and allows for others to retrain it for their own use-case. In fact, a large number of researchers have validated the accuracy of VMAF using their own subjective datasets.

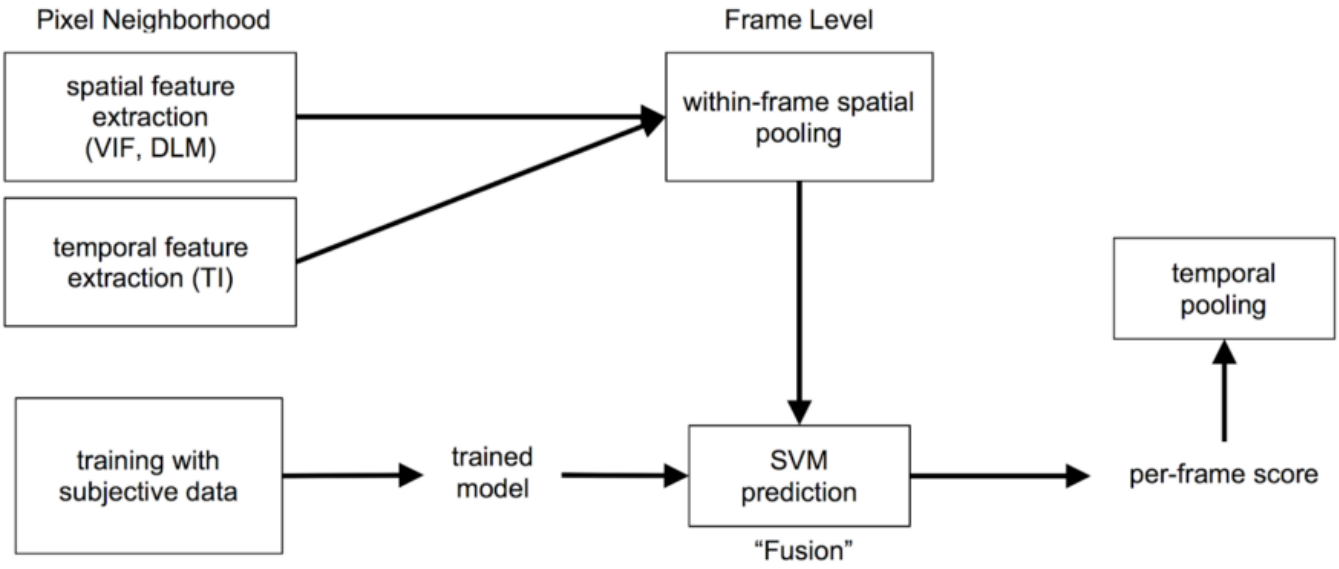


Fig. 2: How VMAF works: pixel level data are pooled to create frame-level features; different spatial and temporal features are fused using SVM regression to create frame-level quality score; consecutive frame scores are pooled to produce final video sequence VMAF score

Convex hull of Rate-Distortion curve

A seminal paper by Ortega and Ramchandran [12] in 1998 showed how to address optimality when dealing with multiple choices in image and video coding.

Assuming that an image consists of N units that need to be coded

- You encode each unit separately, obtaining a (rate, distortion) pair for each possibility, called “operating point”
- You place all available operating points on a rate-distortion graph
- You extract its convex hull, the shell that outlines its boundary
- You pick one point on the convex hull from each unit such that points from different units have (roughly) equal distortion-rate slope.

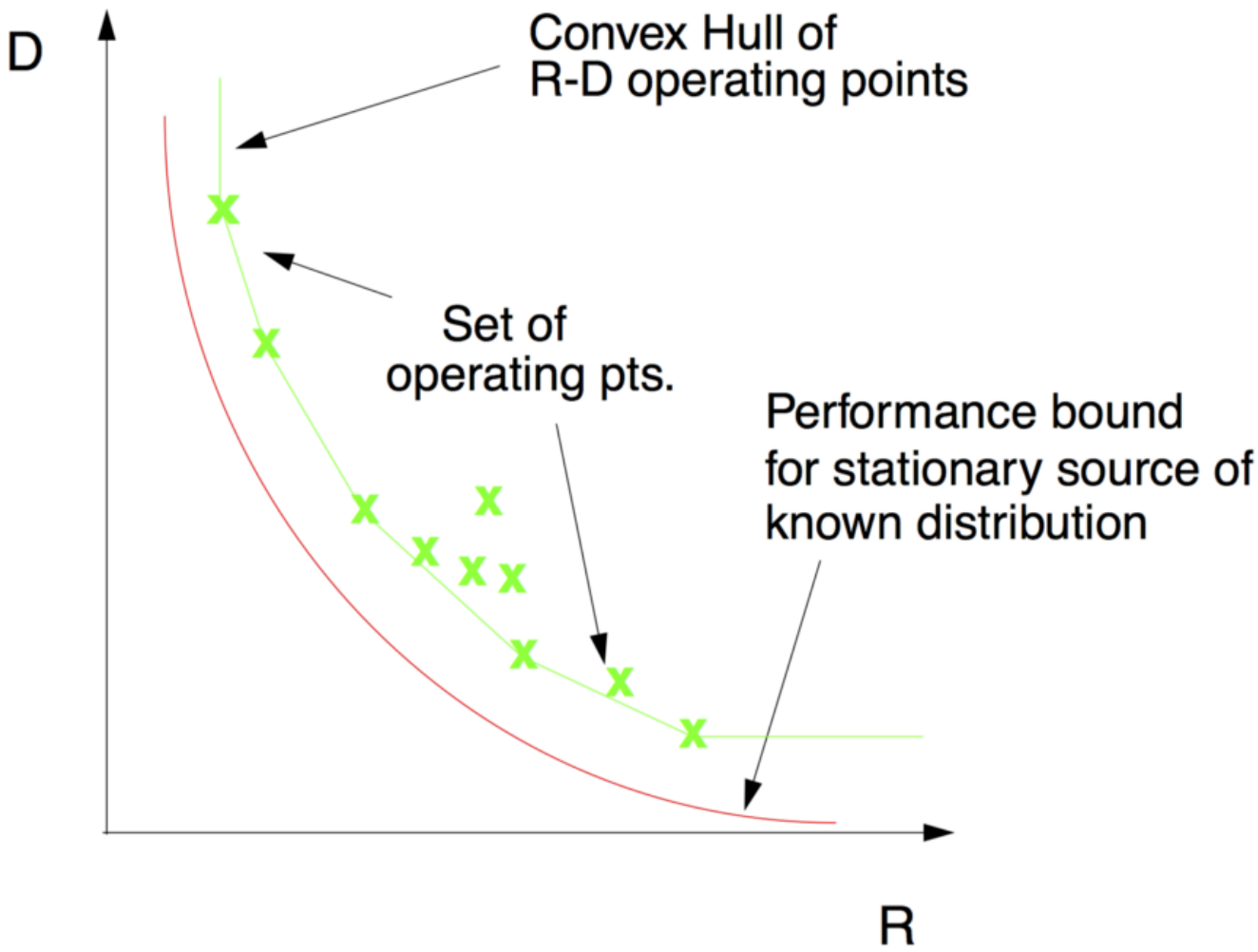


Fig. 3: Example of operational points and their R-D convex hull, together with source R-D curve. Reproduced from [12]

Putting it together

One can thus consider the following system:

- A long video sequence is split in shots

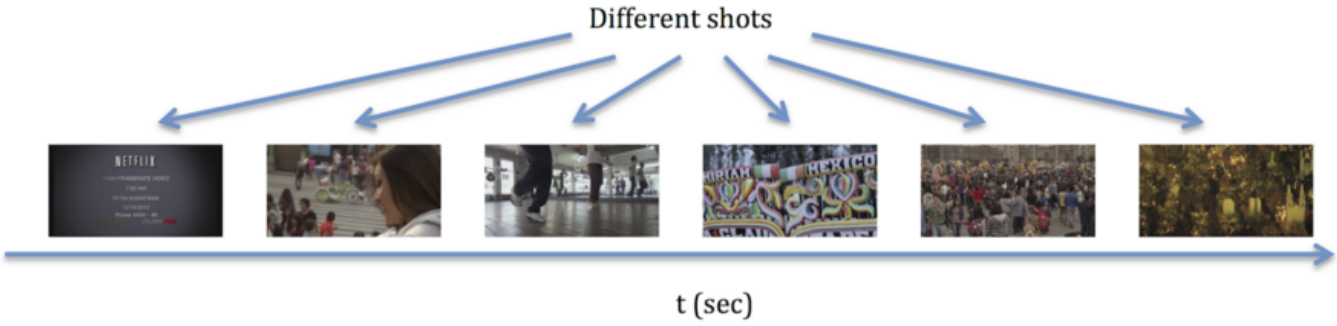


Fig. 4: Thumbnails from representative shots from “El Fuente” test sequence; this title consists of 96 shots

- Each shot is encoded multiple times with different encoding parameters, such as resolutions and qualities (QPs)
- Each encode is evaluated using VMAF, which together with its bitrate produces an (R,D) point. One can convert VMAF quality to distortion using different mappings; we tested against the following two, linearly and inversely proportional mappings, which give rise to different temporal aggregation strategies, discussed in the subsequent section

$$D_{Linear}(VMAF) = 100 - VMAF$$

$$D_{Inverse}(VMAF) = \frac{1}{1+VMAF}$$

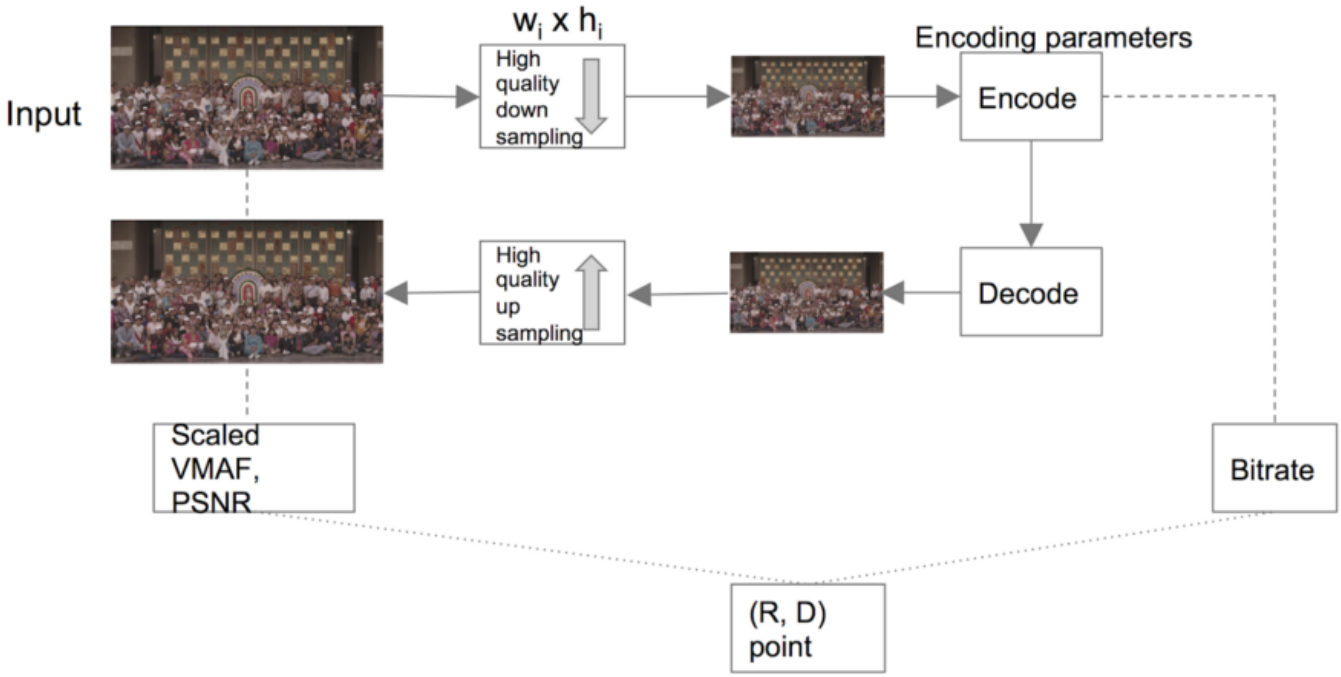


Fig. 5: Encoding a shot using a set of parameters, such as resolution and QP, and obtaining a single (R,D) point for it.

- The convex hull of (R,D) points for each shot is calculated. In the following example figures, distortion is inverse of (VMAF+1)

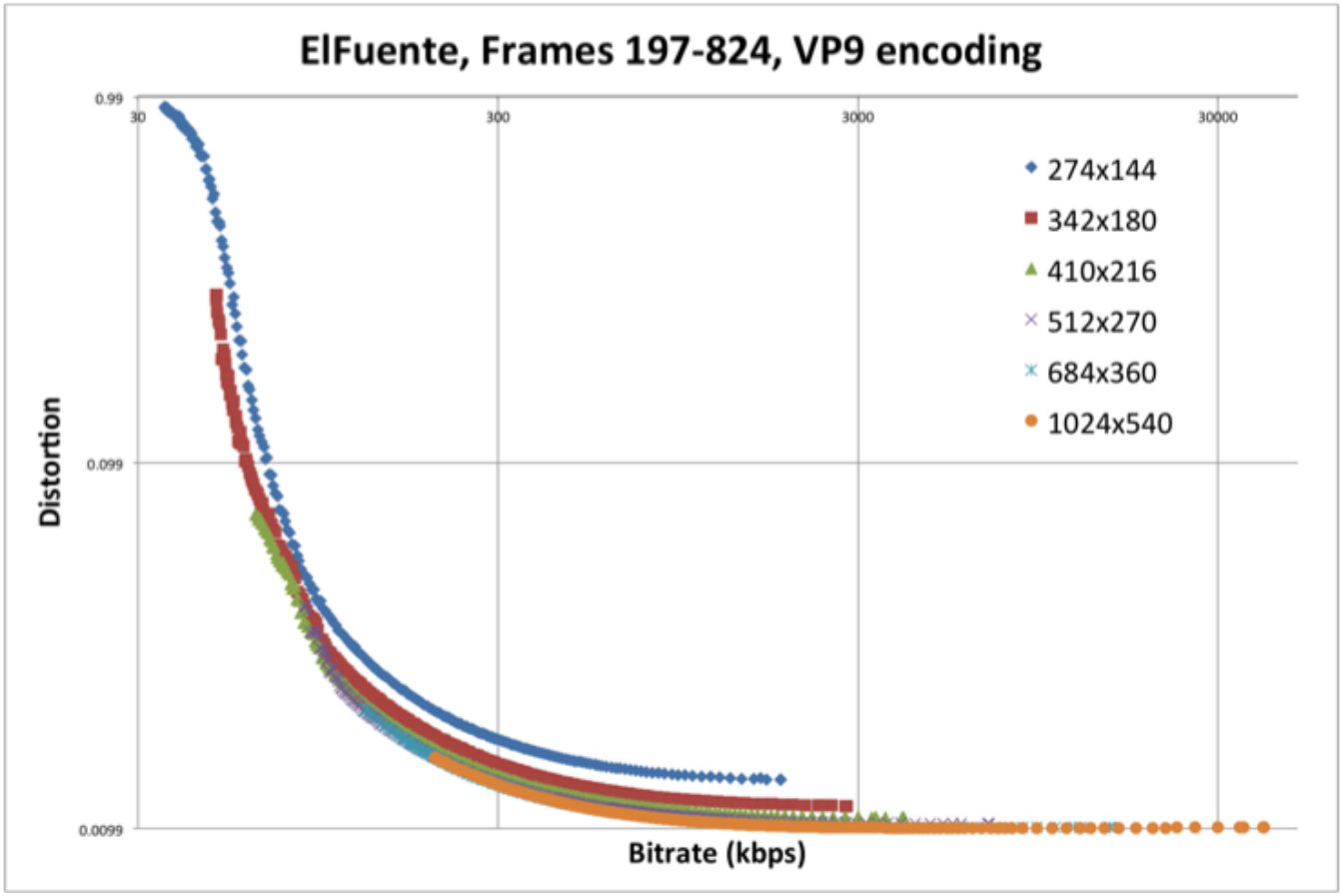


Fig. 6: Multiple (R,D) points for a certain shot from “El Fuente”, obtained at various encoding resolutions and QPs using VP9 (libvpx)

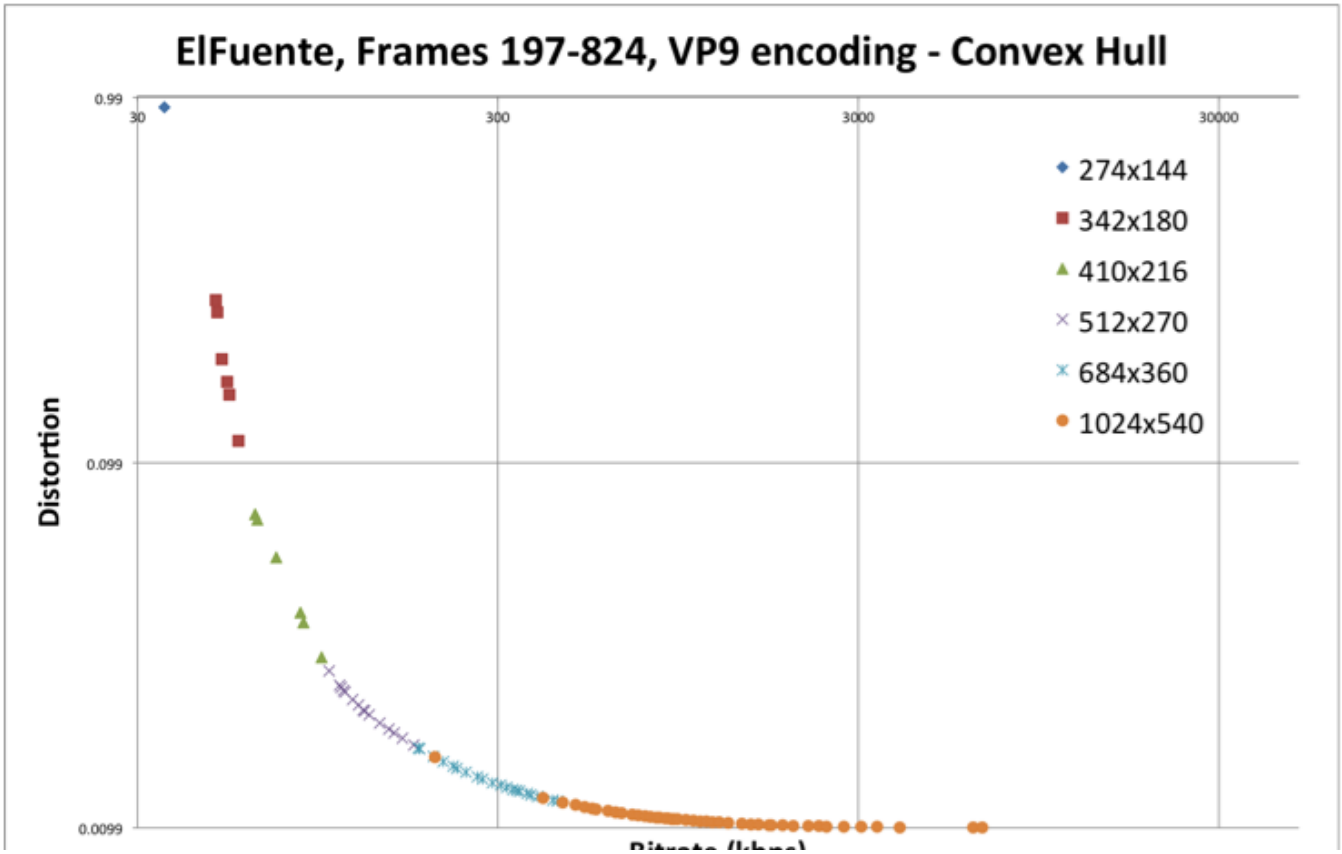


Fig. 7: Convex hull of (R, D) operating points for the same shot from “El Fuente” using VP9 (libvpx)

- Points from the convex hull, one from each shot, are combined to create an encode for the entire video sequence by following the constant-slope principle and building end-to-end paths in a Trellis

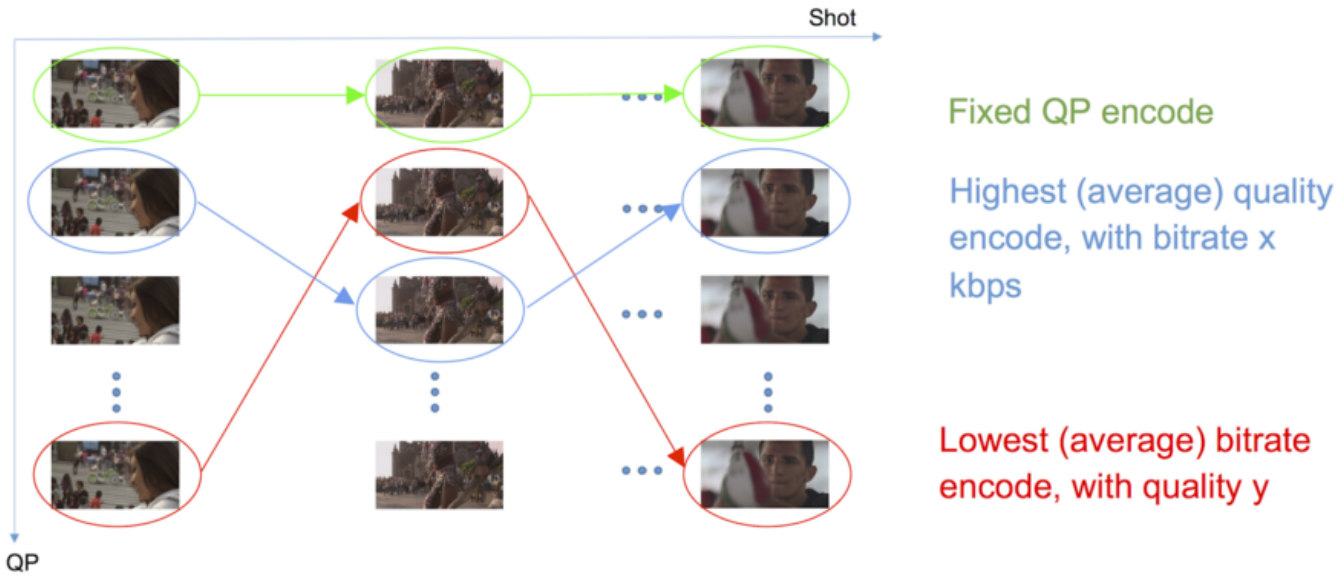
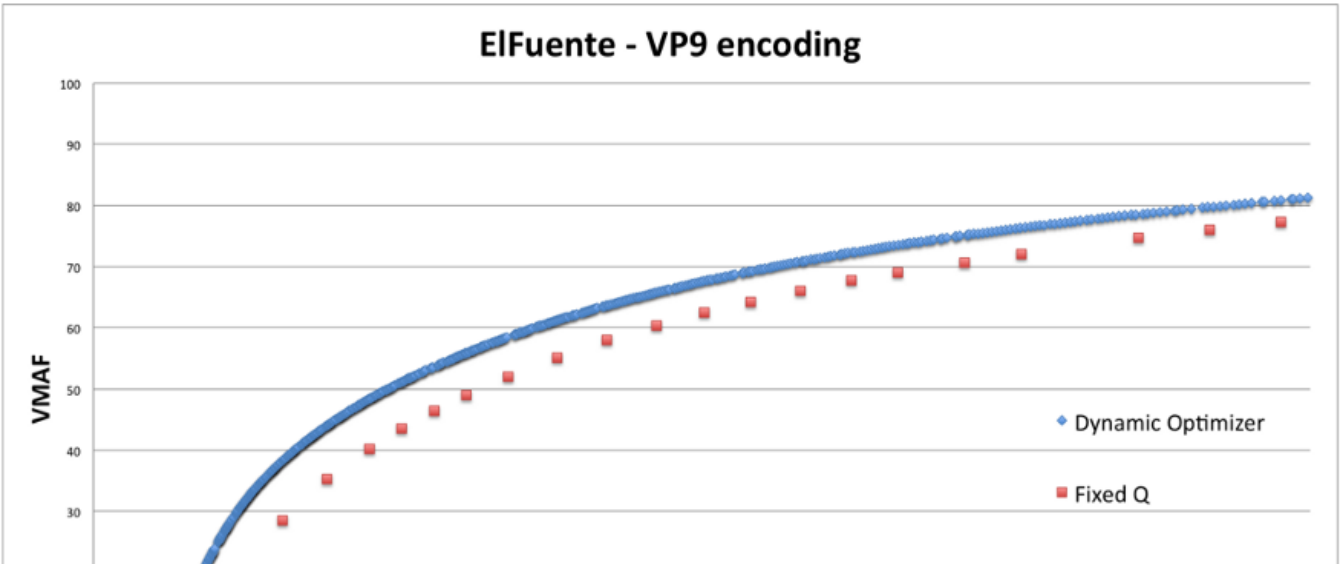


Fig. 8: Combining shot encodes to produce optimal encodes; example Trellis paths show fixed QP encoding, minimizing bitrate for a given average quality or maximizing quality for a given average bitrate. Selected shot encodes have approximately equal slope in (R,D) space.

- One produces as many aggregate encodes (final operating points) by varying the slope parameter of the R-D curve as necessary in order to cover a desired bitrate/quality range

- Final result is a complete R-D or rate-quality (R-Q) curve for the entire video sequence



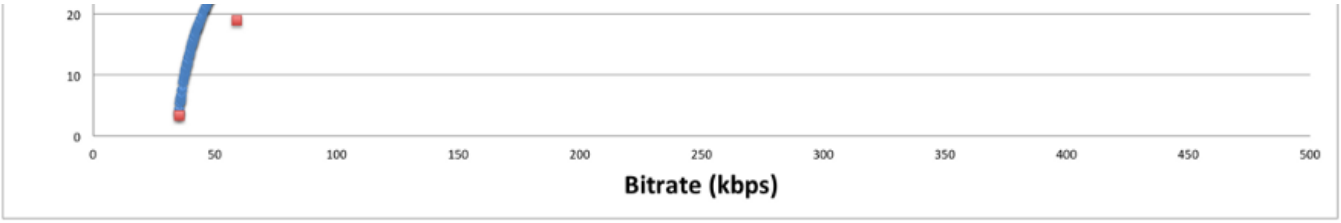


Fig 9: Final R-Q curve obtained for the entire “El Fuente” video sequence. Baseline is the best possible fixed-QP encoding; dynamic optimizer reduces bitrate by an average of 30% in this case

This complete system is called “Dynamic Optimizer” and the framework produces Netflix’s newest generation of encodes.

Testing methodology — Results

10 representative titles from the Netflix catalog were selected and encoded using the VP9-libvpx video codec.

In terms of temporal aggregation, we have implemented various pooling methods, two of them corresponding to the quality-to-distortion mappings introduced earlier, i.e. linear and inversely proportional mapping. We refer to them as arithmetic mean average VMAF (LVMAF) and harmonic mean averaged VMAF (HVMAF).

$$LVMAF = \frac{1}{N}\sum_{i=0}^{N-1} VMAF_i$$

$$HVMAF = \frac{N}{\sum_{i=0}^{N-1}\frac{1}{1+VMAF_i}} - 1$$

These two methods, LVMAF and HVMAF temporal quality aggregation, produced very high quality encoded sequences — allowing for more aggressive or more conservative temporal quality fluctuations in the combined video sequence, respectively.

VP9 encoding recipe

Encoding parameters used in VP9-libvpx were taken from a previous study; its findings were presented at Netflix’s “Open house on royalty-free codecs” held in Oct. 2016. Based on that study, the best configuration to use is “fixed-QP, AQ-mode=0, CPU=0,

best", shown to produce highest quality both in terms of PSNR and VMAF quality metrics. We reproduce key results from that study in the Appendix.

We compared results obtained by the dynamic optimizer against the best possible fixed-QP VP9-libvpx encoding. The methodology followed and various parameters chosen for this experiment are summarized in the following table.

Parameter	**Value**
Video codec	VP9-libvpx
Encoding mode	Shot-based
Encoding recipe	Fixed-QP, AQ-mode=0, CPU=0, best
Quality metric - aggregation strategy	Harmonic VMAF (HVMAF) @ 960x540 display resolution
Baseline	Best-possible fixed-QP encode (convex hull of all fixed-QP encodes in all resolutions)
Dynamic optimizer parameter space	6 resolutions (256x144 - 960x540), 64 QPs per resolution
Video sequences tested	10 full titles
Operating points used for bitrate savings & quality improvement figures	Around 256kbps

Table 1: Parameters used for Dynamic Optimizer Experiment 1

The corresponding gains obtained by the dynamic optimizer, expressed both in terms of % bitrate savings at the same visual quality and in terms of improvement in HVMAF scores at the same average bitrate, are as follows:

TITLE	Best fixed-QP encoding		Dynamic Optimizer @ **same quality**	Dynamic Optimizer @ **same bitrate**
	Bitrate (kbps)	HVMAF (0-100)	Bitrate savings (%)	Delta HVMAF (0-100)
Bloodline	245	86.6	-15%	+2.0
BoJack	230	95.5	-14%	+1.1
Breaking Bad	251	91.8	-16%	+1.7
Marvel's Daredevil	247	92.0	-21%	+1.9

El Fuente	262	36.1	-38%	+21.8
House of Cards	213	92.3	-17%	+1.3
Meridian	259	96.0	-13%	+0.6
Orange is the new black	256	86.2	-11%	+1.6
The Avengers	278	82.0	-18%	+3.4
Wet Hot American Summer	231	78.7	-8%	+1.6
AVERAGE	**247**	**83.7**	**-17.1%**	**+3.7**

Table 2: Experiment 1 Results. R-D performance comparison around 256kbps between dynamic optimizer and best possible fixed-QP encoding for 10 representative titles from the Netflix catalog.

The result was an average bitrate savings of 17.1% over the best possible fixed-QP encoding of the entire video sequence when using HVMAF as quality metric. The improvement when using PSNR is even higher: 22.5% bitrate savings on average.

In this comparison, computational complexity remained constant between the baseline and dynamic optimizer results, since obtaining the convex hull of fixed-QP encodes for an entire sequence requires the same complexity as that for the dynamic optimizer. Thus, this represents a lower-bound on the amount of improvement introduced by the dynamic optimizer.

If we use a more common baseline, such as the 2-pass VBR configuration with CPU=1, good, AQ-mode=2 encoding recipe in VP9-libvpx, the improvement by the dynamic optimizer is much larger: over 50% bitrate savings on average, in terms of HVMAF. One needs to keep in mind, although, that computational complexity of the dynamic optimizer solution is much higher in that case.

How good is it for other video codecs?

Based on what was presented earlier, one can immediately understand that there is nothing codec-specific in the dynamic optimizer framework. In order to confirm this, a set of shorter clips were encoded with H.264/AVC, HEVC and VP9-libvpx, with the following experimental set-up:

Parameter	Value
Video codec	AVC-High/x264, VP9/libvpx, HEVC/x265
Encoding mode	Shot-based
Encoding recipe	AQ-mode=0, fixed QP/CRF, CPU speed varying by resolution
Quality metric - aggregation strategy	Linear VMAF (LVMAF) @ 1920x1080 display resolution
Baseline	Fixed-QP encode (middle QP value for each resolution)
Dynamic optimizer parameter space	7 resolutions (384x216 - 1920x1080), 7 QPs per resolution
Video sequences tested	15 min. clip x 35 titles from Netflix catalog
Operating points used for bitrate savings & quality improvement figures	BD-rate across entire quality range

Table 3: Parameters used for Dynamic Optimizer Experiment 2

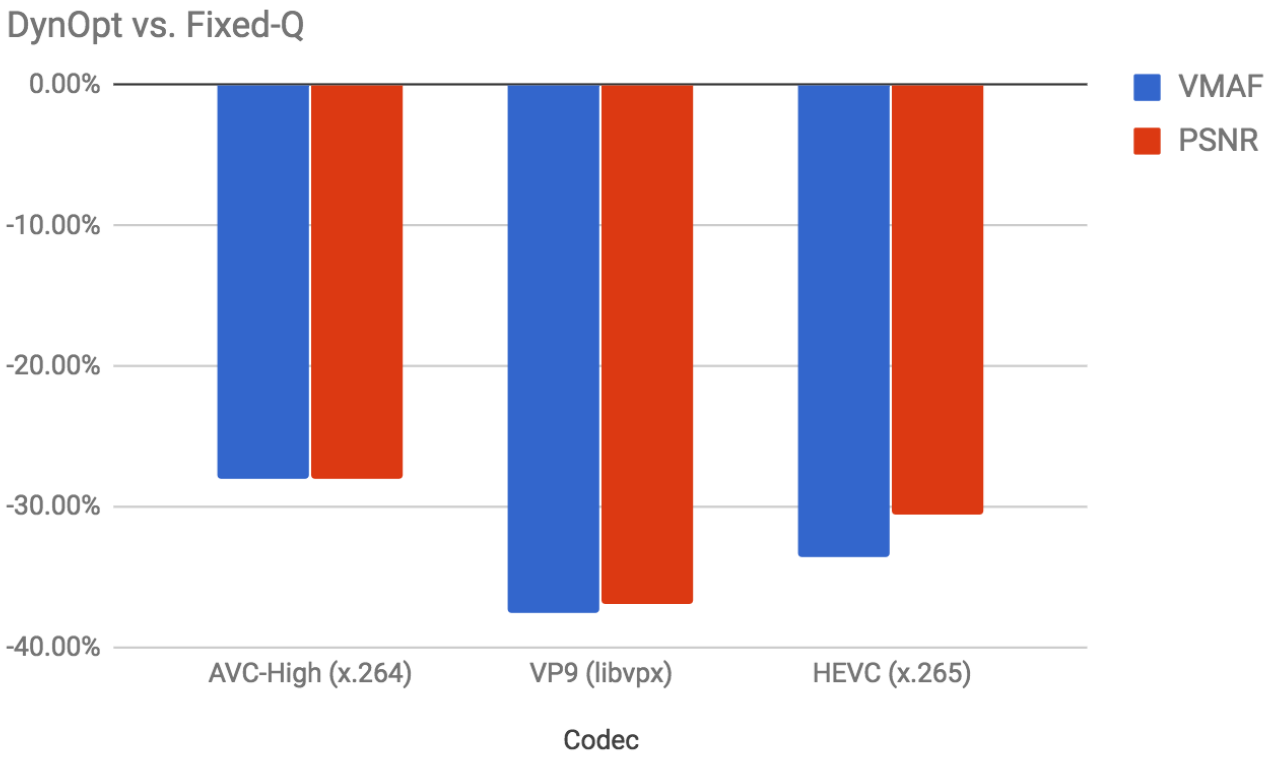


Fig. 10: Experiment 2 Results. BD-rate improvement by the dynamic optimizer over fixed-QP/CRF encoding for AVC-High (x264), VP9 (libvpx) and HEVC (x265) in terms of VMAF and PSNR.

Codec	VMAF	PSNR
AVC-High (x264)	-28.04%	-27.99%
VP9 (libvpx)	-37.61%	-36.97%
HEVC (x265)	-33.51%	-30.52%

Table 4: Experiment 2 Results. BD-rate improvement by the dynamic optimizer over fixed-QP/CRF encoding using different video codecs and quality metrics

One can notice that the dynamic optimizer improves all three codecs by approximately 28–38%. Keep in mind that these improvements are not comparing performance between codecs but rather how each one of these codecs can be improved by using the dynamic optimizer framework. A more thorough comparison of state-of-the-art video codecs, using the dynamic optimizer as high-level encoding framework, will be published in the upcoming weeks.

Dynamic optimizer summary

Dynamic optimizer is an optimization framework for video encoding. Its key features are the following:

- Shot-based encoding
- Multiple encoding at different resolutions and quality parameters
- Perceptual assessment and tuning of quality by using VMAF as its core metric
- Massively parallel processing, ideal for cloud-based video encoding software pipelines

Its key advantages are the following:

- It can be applied to any existing or future video codec, which qualifies it as a video encoding optimization framework
- It can help future codec development by identifying “perceptually relevant” ranges of encoding resolutions and qualities (QPs) for each test video sequence, which can be used while developing and evaluating performance of new coding tools

- It removes much of the rate-control factor in video codec implementations, thus allowing for much more fair comparisons of video codecs
- It is orthogonal to improvements one can bring in the shot-encoding recipe, such as better I-B-P coding structure, spatially adaptive QP selection; any improvements performed at the shot level are additive to those brought by the dynamic optimizer
- Its complexity can be scaled up or down, depending on the amount of compute resources, offering a trade-off between complexity and rate-distortion efficiency
- It produces fully compliant bitstreams
- It can be used with VMAF, PSNR or any other video quality metric.
- It benefits both standalone bitstream creation, intended for downloading and offline consumption, as well as full bitrate ladder creation, used for adapting streaming

Cloud Implementation

We've implemented the dynamic optimizer framework in our encoding pipeline, leveraging our scalable cloud infrastructure and under-utilized cloud instances during non-peak streaming hours [13],[14]. We've applied this encoding system to AVC-High and VP9 streams, improving our members' video quality as well as saving bandwidth. Stay tuned for another tech blog describing our implementation and results!

Acknowledgement

This work is the collective result of the entire Video Algorithms team at Netflix. I would like to personally thank Anne Aaron, Chao Chen, Jan De Cock, Rich Gerber, Liwei Guo, Zhi Li, Megha Manohara, Aditya Mavlankar, Anush Moorthy, Andrey Norkin, Kyle Swanson and David Ronca for all their contributions.

References

[1] ISO/IEC 11172–2:1993 “Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 2: Video”

[2] ISO/IEC 23008–2:2013 “Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: Video”

[3] ISO/IEC 14496–10:2014 “Information technology — Coding of audio-visual object — Part 10: Advanced Video Coding”

[4] ISO/IEC 13818–2:2013 “Information technology — Generic coding of moving pictures and associated audio information — Part 2: Video”

[5] J. De Cock, A. Mavlankar, A. Moorthy and A. Aaron, “A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications”, Proc. of the SPIE 9971, Applications of Digital Image Processing XXXIX, 997116 (27 Sep. 2016)

[6] A. Grange, P. de Rivaz, and J. Hunt, “VP9 Bitstream and Decoding Process Specification”, Google, 2016

[7] “Adaptive bitrate streaming”, Wikipedia — The Free Encyclopedia, https://en.wikipedia.org/wiki/Adaptive_bitrate_streaming

[8] A. Aaron and D. Ronca, “High quality video encoding at scale,” The NETFLIX tech blog, Dec. 9, 2015, link: http://techblog.netflix.com/2015/12/high-quality-video-encoding-at-scale.html

[9] A. Aaron, Z. Li, M. Manohara, J. De Cock and D. Ronca, “Per-title encode optimization”, The NETFLIX tech blog, Dec. 14, 2015, link: http://techblog.netflix.com/2015/12/per-title-encode-optimization.html

[10] A. Norkin, J. De Cock, A. Mavlankar and A. Aaron, “More Efficient Mobile Encodes for Netflix Downloads”, The NETFLIX tech blog, Dec. 1, 2016, link: https://medium.com/netflix-techblog/more-efficient-mobile-encodes-for-netflix-downloads-625d7b082909

[11] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, “Toward a practical perceptual video quality metric,” The NETFLIX tech blog, June 5, 2016, link: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html

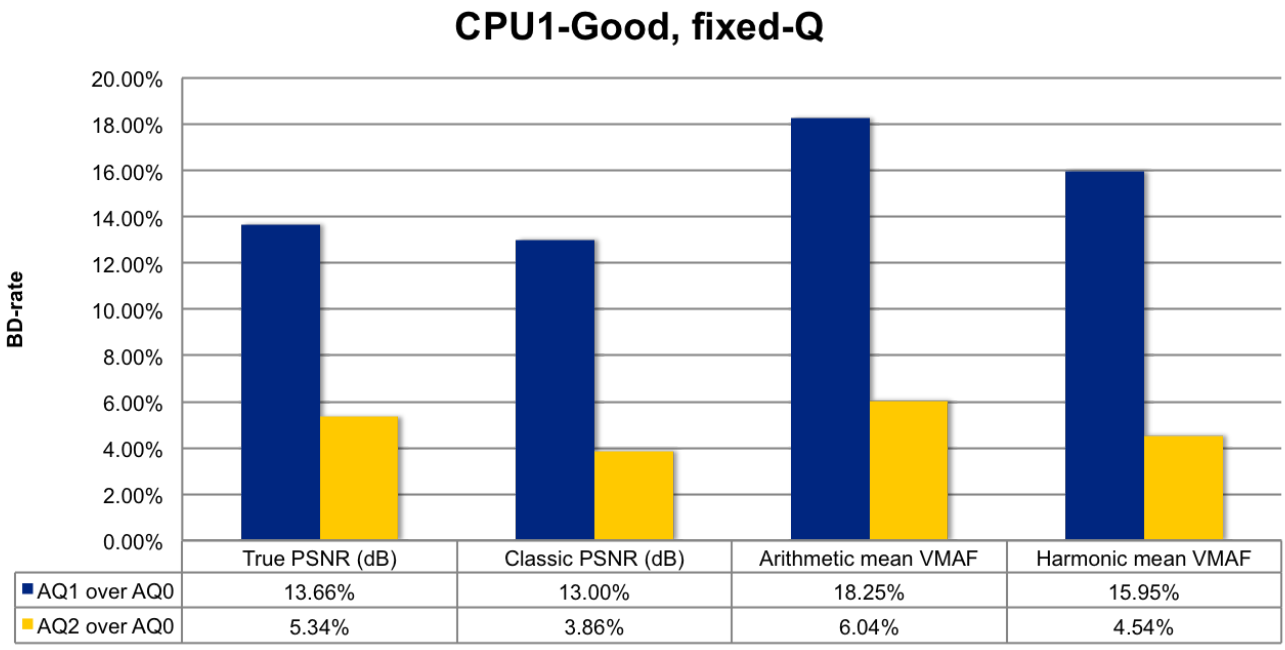
[12] A. Ortega and K. Ramchandran, “Rate-distortion methods for image and video compression: An overview,” *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998

[13] A. Park, D. Derlinger and C. Watson “Creating your own EC2 spot market,” The NETFLIX tech blog, Sep. 28, 2015, link: http://techblog.netflix.com/2015/09/creating-your-own-ec2-spot-market.html

[14] R. Wong, D. Derlinger, A. Shiroor, N. Mareddy, F. San Miguel, R. Gallardo and M. Prabhu “Creating your own EC2 spot market — part 2,” The NETFLIX tech blog, Nov. 23, 2015, link: http://techblog.netflix.com/2015/11/creating-your-own-ec2-spot-market-part-2.html

Appendix: VP9 recipe testing

Encoding parameters used in VP9-libvpx were taken from a previous study; its findings were presented at Netflix’s “Open house on royalty-free codecs” in Oct. 2016. Based on that study, which used the same set of 10 full-titles for testing as those chosen for the first experiment reported earlier, the best configuration to use is “fixed-QP, AQ-mode=0, CPU=0, best”, shown to produce highest quality both in terms of PSNR and VMAF quality metrics. The following figures show the effect in terms of average BD-rate loss when choosing different parameters in VP9-libvpx encoding.



	True PSNR (dB)	Classic PSNR (dB)	Arithmetic mean VMAF	Harmonic mean VMAF
AQ1 over AQ0	13.66%	13.00%	18.25%	15.95%
AQ2 over AQ0	5.34%	3.86%	6.04%	4.54%

Fig. 11: Effect of AQ-mode=0/1/2 in VP9-libvpx encoding; AQ-mode=0 yields better results in all metrics (from Oct. 2016 presentation on "VP9 Encoding Opportunities @ Netflix", part of Netflix's "Open House on Royalty Free Codecs")

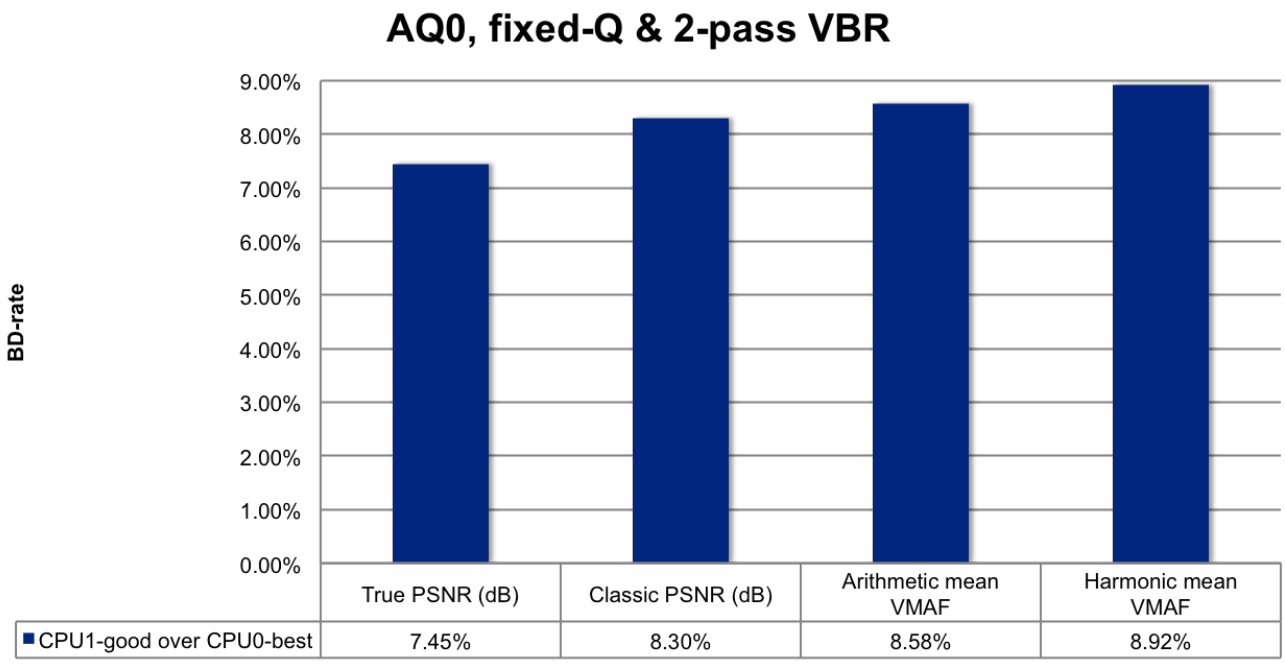


Fig. 12: Effect of CPU0/best in VP9-libvpx encoding; CPU0/best is the slowest, but provides consistently better results than other settings (from Oct. 2016 presentation on "VP9 Encoding Opportunities @ Netflix", part of Netflix "Open House on Royalty Free Codecs")

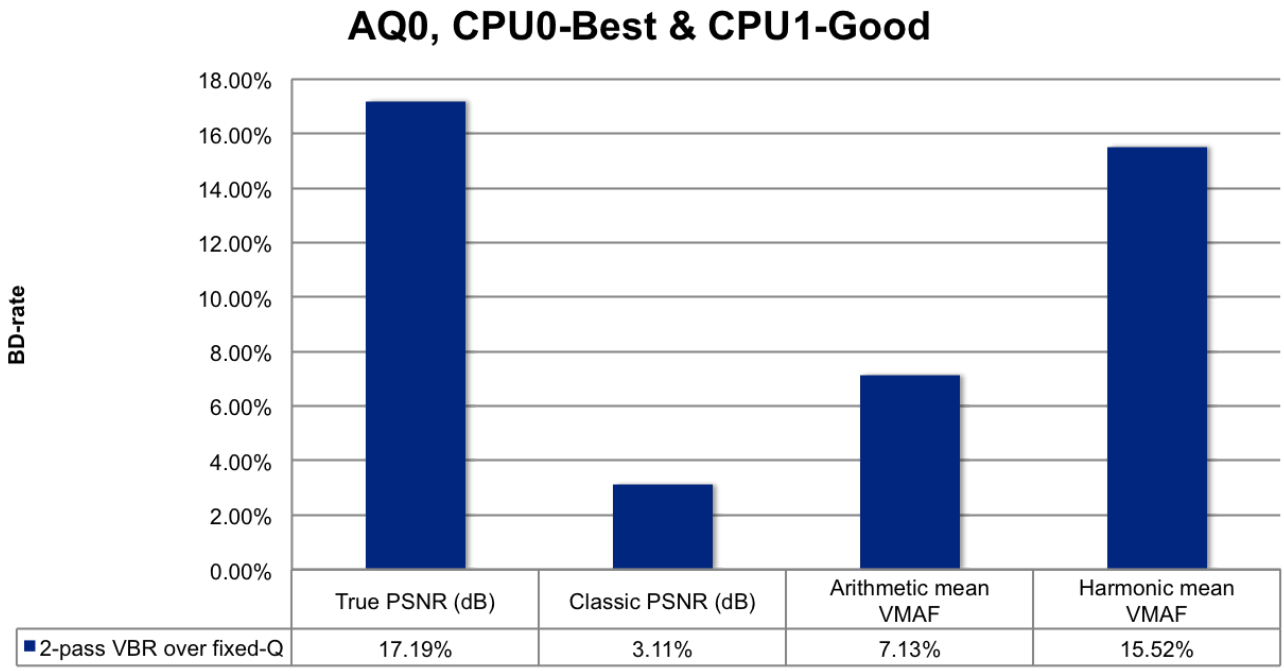


Fig. 13: BD-rate loss of 2-pass VBR over fixed-QP encoding in VP9-libvpx (from Oct. 2016 presentation on "VP9 Encoding Opportunities @ Netflix", part of Netflix's "Open House on Royalty Free Codecs")

Video Encoding Optimization Algorithms Video Quality Netflix Streaming