On VMAF’s Property in the Presence of Image Enhancement Operations

Zhi Li, Netflix

VQEG Fall 2020
Outline

- Introduction
  - Emergence of new use cases
  - VMAF in codec evaluation
  - Our position
- Foundations: VMAF’s property
  - Visual Information Fidelity (VIF)
  - Detail Loss Metric (DLM)
- Proposed modifications
- Results
  - Image enhancement operations
  - Libaom tune=vmaf mode
  - Correlation on compression/scaling public datasets
  - VMAF monotonicity with quantization
  - VMAF vs. VMAF NEG
VMAF and image enhancement

- The origin of VMAF
  - Video quality of professionally generated movies and TV shows
  - Adaptive streaming
    - Compression artifacts
    - Scaling artifacts

- Emerging new use cases
  - UGC, Gaming, VR
  - Quite common to include image enhancements
    - Sharpening
    - Contrasting
    - Histogram equalization
    - ...

- VMAF-driven video enhancement and encoding
  - MSU paper
  - libaom tune=vmaf mode
**Original**  
VMAF 97.4277

**Sharpening**  
VMAF 111.9868*

**Histogram Equalization**  
VMAF 144.0195*

*By default, VMAF score is clipped between [0, 100] in the last step. Here the clipping is disabled using the option `disable_clip`. 
VMAF in codec evaluation

- Desirable to measure gain from compression without measuring pre-processing
- Difficult to strictly separate encoder from pre-processing steps
  - Especially for proprietary encoders
- It may become difficult to use VMAF to assess pure compression gain
“The video looks better, sure, but you could have/should have achieved the same impact by optimizing contrast before encoding.”

Doesn't this realization contradict the claim that VMAF can be hacked. VMAF measures perceptual quality which cannot be assessed by SSIM, so it’s not necessary to observe the same trends between the two metrics. An experiment that you could do in your article would be to conduct some crowd sourced MOS survey (e.g. through Amazon Turk) to illuminate whether VMAF increases in line with MOS for those videos. If VMAF aligns with MOS but SSIM doesn’t it means that it’s not hacking, or at least it means that human perception of visual quality is hackable which is something that video encoding should use.

Good point, and understood, and that’s why I gauged BitSave as a valid technology. However, as I showed with the table, there are times where increasing contrast darkens the video and makes it look noticeably worse, though the VMAF score is improved.

And yes, subjective observations are the gold standard which is why I say in my Streaming Media article, ‘After many hours of testing, I found that BitSave’s technology is valid and valuable, though the proof of the pudding will be how it performs in subjective testing with your test clips. Subjective evaluations of the BitSave clips would have been great, but was outside the time and expense budget for the review.'
First ever VMAF meme

https://www.reddit.com/r/AV1/comments/g19ary/more_vmaf_more_better
Our position

● There is value for VMAF to disregard image enhancement gain that is not part of the codec
● There is also value for VMAF to preserve the measure of image enhancement gain to reflect quality perceived by end users
● Solution
  ○ Introduce knobs in VMAF to control the measured enhancement gain
  ○ Currently, two models:
    ■ Default model
    ■ NEG ("No Enhancement Gain") model
● Recommendation
  ○ Use NEG model for codec evaluation
  ○ Use default model to assess compression and enhancement combined
    ■ In future versions, we will address overprediction issue related to overusing (abusing) of image enhancement operations
Foundations
VMAF framework

*VMAF stands for Video Multi-method Assessment Fusion*
Visual Information Fidelity (VIF)

\[ U \sim N(0, \sigma_U^2) \]

Source: \( C \sim N(0, s^2\sigma_U^2) \)

Distorted: \( D \)

Perceived Source: \( E \)

Gain: \( g \)

Additive Noise: \( V \sim N(0, \sigma_V^2) \)

\[ \sigma_V^2 = \sigma_D^2 - g \cdot \sigma_{CD} \]

\[ N \sim N(0, \sigma_N^2) \]

\[ N' \sim N(0, \sigma_N^2) \]
Visual Information Fidelity (VIF) - Cont’d

\[ VIF_\lambda = \frac{N}{\sum_{i=1}^{N} \log_2 \left( 1 + \frac{g_i^4 s_i^2 \sigma_u^2}{\sigma_v_i^2 + \sigma_N^2} \right)} \]

- \( g = \frac{\sigma_{CD}}{\sigma_C^2} \)
- \( \sigma_v^2 = \sigma_D^2 - g \cdot \sigma_{CD} \)

\( s \): pixel position; \( \lambda \): scale (1, 2, 3, 4)
**Detail Loss Measure (DLM)**

\[
R = \text{clip}_{[0,1]} \left( \frac{T}{O} \right) \cdot O
\]

\[
R = T, \text{ if } |\psi_O - \psi_T| < 1^\circ
\]

\[
DLM = \left( \frac{\sum_{\lambda=1}^{4} \sum_{\theta=2}^{4} \left( \sum_{i,j \in \text{center}} CM(CSF(R(\lambda,\theta,i,j)))^3 \right)^{\frac{1}{3}}}{\sum_{\lambda=1}^{4} \sum_{\theta=2}^{4} \left( \sum_{i,j \in \text{center}} CM(CSF(O(\lambda,\theta,i,j)))^3 \right)^{\frac{1}{3}}} \right)^{\frac{1}{3}}
\]

\(i, j\): pixel position; \(\lambda\): scale (1, 2, 3, 4); \(\theta\) subbands (1, 2, 3, 4)

Wavelet coefficients:
- \(O\): original (source)
- \(T\): target (distorted)
- \(R\): restored
- \(A\): additive
Proposed Modifications
Visual Information Fidelity (VIF) - Cont’d

\[ g_i = \min(g_i, EGL_{VIF}), \text{ where } EGL_{VIF} \geq 1.0 \]

\[
VIF_{\lambda} = \frac{\sum_{i=1}^{N} \log_2 \left( 1 + \frac{g_i^2 s_i^2 \sigma_u^2}{\sigma_{v_i}^2 + \sigma_N^2} \right)}{\sum_{i=1}^{N} \log_2 \left( 1 + \frac{s_i^2 \sigma_u^2}{\sigma_N^2} \right)}
\]

\[ g = \frac{\sigma_{CD}}{\sigma_C^2} \]

\[ \sigma_{V}^2 = \sigma_D^2 - g \cdot \sigma_{CD} \]

\[ EGL_{VIF}: \text{ VIF enhancement gain limit} \]
Detail Loss Measure (DLM)

\[ R = \text{clip}_{[0,1]} \left( \frac{T}{O} \right) \cdot O \]

\[ R = T, \text{ if } |\psi_O - \psi_T| < 1^\circ \]

\[ R = \min (R \square EGL_{DLM}, T), \text{ if } |\psi_O - \psi_T| < 1^\circ \text{ and } R > 0; \]

\[ R = \max (R \square EGL_{DLM}, T), \text{ if } |\psi_O - \psi_T| < 1^\circ \text{ and } R < 0, \]

where \( EGL_{DLM} \geq 1.0. \)

\( EGL_{DLM}: \) DLM enhancement gain limit
Summary of modifications

- Introduce two knobs
  - VIF enhancement gain limit $EGL_{VIF} \geq 1.0$
  - DLM enhancement gain limit $EGL_{DLM} \geq 1.0$
- For default model
  - Set both limits to a large value (example: 100.0)
- For NEG (“No Enhancement Gain”) model
  - Set both limits to 1.0
- Future work
  - Future models will provide standard values for these limits
Results
Pure image enhancement operations

Original
VMAF 97.4277
VMAF NEG 97.4280

Sharpening
VMAF 111.9868*
VMAF NEG 85.3330

Histogram Equalization
VMAF 144.0195*
VMAF NEG 78.7122

*By default, VMAF score is clipped between [0, 100] in the last step. Here the clipping is disabled using the option disable_clip.
libaom encoding

Original
VMAF 97.4277
VMAF NEG 97.4280

libaom CQ 43
VMAF 95.1425
VMAF NEG 93.4151

Libaom tune=vmaf CQ 43
VMAF 104.8277*
VMAF NEG 87.6951

*By default, VMAF score is clipped between [0, 100] in the last step. Here the clipping is disabled using the option disable_clip.
## BD rate: libaom vs. libaom tune=vmaf 540p

<table>
<thead>
<tr>
<th>Sequence</th>
<th>CPSNRY</th>
<th>TPSNRYUV</th>
<th>VMAF</th>
<th>VMAF NEG</th>
<th>SSIM</th>
<th>MSSSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOTA2_60f_420</td>
<td>746.605</td>
<td>452.357</td>
<td>-55.745</td>
<td>48.116</td>
<td>57.826</td>
<td>46.032</td>
</tr>
<tr>
<td>MINECRAFT_60f_420</td>
<td>76.24</td>
<td>77.044</td>
<td>-26.241</td>
<td>13.359</td>
<td>41.991</td>
<td>26.887</td>
</tr>
<tr>
<td>Netflix_Aerial_1920x1080_60fps_8</td>
<td>209.234</td>
<td>200.622</td>
<td>-38.326</td>
<td>55.3</td>
<td>85.949</td>
<td>40.925</td>
</tr>
<tr>
<td>Netflix_Boat_1920x1080_60fps_8bi</td>
<td>46.338</td>
<td>44.377</td>
<td>-13.049</td>
<td>21.354</td>
<td>33.888</td>
<td>12.438</td>
</tr>
<tr>
<td>Netflix_Crosswalk_1920x1080_60fp</td>
<td>44.529</td>
<td>43.662</td>
<td>-35.379</td>
<td>9.403</td>
<td>13.461</td>
<td>11.632</td>
</tr>
<tr>
<td>Netflix_FoodMarket_1920x1080_60f</td>
<td>155.067</td>
<td>147.682</td>
<td>-37.414</td>
<td>43.302</td>
<td>43.733</td>
<td>35.534</td>
</tr>
<tr>
<td>Netflix_PierSeaside_1920x1080_60</td>
<td>-</td>
<td>-</td>
<td>-61.555</td>
<td>203.436</td>
<td>248.369</td>
<td>140.171</td>
</tr>
<tr>
<td>Netflix_SquareAndTimelapse_1920x</td>
<td>71.071</td>
<td>69.108</td>
<td>-19.333</td>
<td>19.872</td>
<td>35.237</td>
<td>23.625</td>
</tr>
<tr>
<td>Netflix_TunnelFlag_1920x1080_60f</td>
<td>-</td>
<td>-</td>
<td>-18.212</td>
<td>108.605</td>
<td>88.316</td>
<td>77.793</td>
</tr>
<tr>
<td>STARCAST_60f_420</td>
<td>289.836</td>
<td>250.835</td>
<td>-53.268</td>
<td>55.257</td>
<td>78.509</td>
<td>48.717</td>
</tr>
<tr>
<td>aspen_1080p_60f</td>
<td>65.496</td>
<td>63.017</td>
<td>-31.592</td>
<td>12.528</td>
<td>18.869</td>
<td>16.457</td>
</tr>
<tr>
<td>ducks_take_off_1080p50_60f</td>
<td>-</td>
<td>1132.993</td>
<td>-44.217</td>
<td>55.551</td>
<td>145.881</td>
<td>97.535</td>
</tr>
<tr>
<td>life_1080p30_60f</td>
<td>667.974</td>
<td>462.457</td>
<td>-46.887</td>
<td>60.56</td>
<td>78.882</td>
<td>55.956</td>
</tr>
<tr>
<td>rush_hour_1080p25_60f</td>
<td>50.529</td>
<td>49.277</td>
<td>-40.064</td>
<td>0.177</td>
<td>12.795</td>
<td>13.188</td>
</tr>
<tr>
<td>wikipedia_420</td>
<td>91.43</td>
<td>77.198</td>
<td>-59.998</td>
<td>36.168</td>
<td></td>
<td>109.07</td>
</tr>
</tbody>
</table>
Prediction accuracy: correlation with public datasets (compression and scaling-only)
VMAF (PSNR) vs. QP: Monotonicity

- VMAF (NEG) can capture small changes in quality with QP (or other coding parameters) just like PSNR

More data points at:
https://drive.google.com/drive/folders/1XwM1Vf0PYEvUF9sSMWRDg3Xa0P-aykim?usp=sharing
VMAF vs. VMAF NEG: 1080p

More data points at: https://drive.google.com/drive/folders/1_xUKe8_Mn-HZjC7MPUGEgBSsPlhyq-kO?usp=sharing
Conclusions

- One unique feature of VMAF that differentiates it from PSNR and SSIM is that VMAF captures visual gain from image enhancements.
- For codec evaluation, it is often desirable to evaluate the pure gain from compression.
- Our recommendations:
  - Use **NEG** model for codec evaluation.
  - Use **default** model to assess compression and enhancement combined.
- In future versions, we will address overprediction issue related to overusing (abusing) of image enhancement operations.
Toward a Better Quality Metric for the Video Community

by Zhi Li, Kyle Swanson, Christos Bampis, Lukáš Krasula and Anne Aaron

Over the past few years, we built a tool not just for Netflix, but it highlights our recent progress.

v2.0.0

kylophile released this 6 days ago. 8 commits to master since this release

(2020-12-4) [v2.0.0]

This is a major release with an updated and overhauled vmaf API. The vmafossexec command line tool has been deprecated and replaced with the more flexible and powerful vmaf tool. For an introduction to the vmaf v2.6.8 API as well as an explanation of the new vmaf tool, please see the following README files: libvmaf, vmaf. Also part of this release is a new fixed-point and 86 SIMD-optimized (AVX2, AVX-512) implementation that acheives ~2x speed up compared to the previous floating-point version.

Memo: https://tinyurl.com/y34mgafa
Tech Blog: https://netflixtechblog.com/toward-a-better-quality-metric-for-the-video-community-7ed94e752a30
libvmaf v2.0.0: https://github.com/Netflix/vmaf/releases/tag/v2.0.0
Backup Slides
Looks completely insane to me just how much better it got at the 1mbit point. How long did it take to encode all 15s? I am getting 1fps at cpu-used 4

MrSmilingWolf 4 points · 4 months ago
It took 12.5h running all three VMAF encodes together.

As a side note, --tune=vmaf_with_preprocessing takes 2.5h for the same encode and gives more or less comparable results, so that might be a better tradeoff.

utack 1 point · 4 months ago
Thanks for letting me know

AutoAltRef6 2 points · 4 months ago
The speed levels are probably going to be pretty close to each other when using this tune. As far as I've been able to determine, the VMAF tune parts are entirely single-threaded and hugely bottleneck the encoder.

shananalla88 2 points · 4 months ago · edited 4 months ago
Purely from a visual perspective, the 500k Tears of Steel vmaf-tune clip looks only slightly worse (IMO) that the 1000k psnr-tune clip.

This is more evidence that the quality-improvements/bit-rate savings mentioned in the commit message are quite big (30-40% at least). I hope they speed this mode up soon so it can be used for practical encodes, and not just testing.

https://www.reddit.com/r/AV1/comments/f0g3yx/libaom_adds_vmaf_tune/
Enhancement gain visualization

(a) Original
VMAF 97.42 VMAF NEG 97.42

(b) Sharpening
VMAF 100 VMAF NEG 85.33

(c) Histogram Equalization
VMAF 100 VMAF NEG 78.71

(d) libaom CQ 43
VMAF 95.14 VMAF NEG 93.41

(e) libaom CQ 43 tune=vmaf
VMAF 100 VMAF NEG 87.69

(f) libaom CQ 43 tune=vmaf
Enhancement Gain Visualization
Analyzing libaom
  tune=vmafneg mode
diff (source, tune=vmf, qp43)
Frame value:
PSNR 44.14
VMAF 95.35
VMAF NEG 93.03

300 frames:
41.64 Kbps

Tune = vmafneg, qp43
Frame value:
PSNR 44.14
VMAF 95.35
VMAF NEG 93.03

300 frames:
41.64 Kbps
diff (source, tune=vmafneg, qp43)
Sequence: akiyo_cif

Blue: tune=vmafneg
Red: tune=psnr
BD-VMAFneg -6.2%
Sequence: akiyo_cif

Blue: tune=vmafneg
Red: tune=psnr
BD PSNR 5.9%