W4: OBJECTIVE QUALITY METRICS 2D/3D

Jan Ozer
www.streaminglearningcenter.com
janozer@gmail.com
276-235-8542
@janozer

Course Overview

- Section 1: Validating metrics
- Section 2: Comparing metrics
- Section 3: Computing metrics
- Section 4: Applying metrics
- Section 5: Using metrics
- Section 6: 3D metrics

Section 1: Validating Objective Quality Metrics

- What are objective quality metrics?
- How accurate are they?
- How are they used?
- What are the subjective alternatives?

What Are Objective Quality Metrics

- Mathematical formulas that (attempt to) predict how human eyes would rate the videos
  - Faster and less expensive than subjective tests
  - Automatable
- Examples
  - Video Multimethod Assessment Fusion (VMAF)
  - SSIMPLUS
  - Peak Signal to Noise Ratio (PSNR)
  - Structural Similarity Index (SSIM)

$$
\begin{aligned}
PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\
&= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\
&= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)
\end{aligned}
$$

Measure of Quality Metric

- Role of objective metrics is to predict subjective scores
- Correlation with Human MOS (mean opinion score)
  - Perfect score - objective MOS matched actual subjective tests
  - Perfect diagonal line

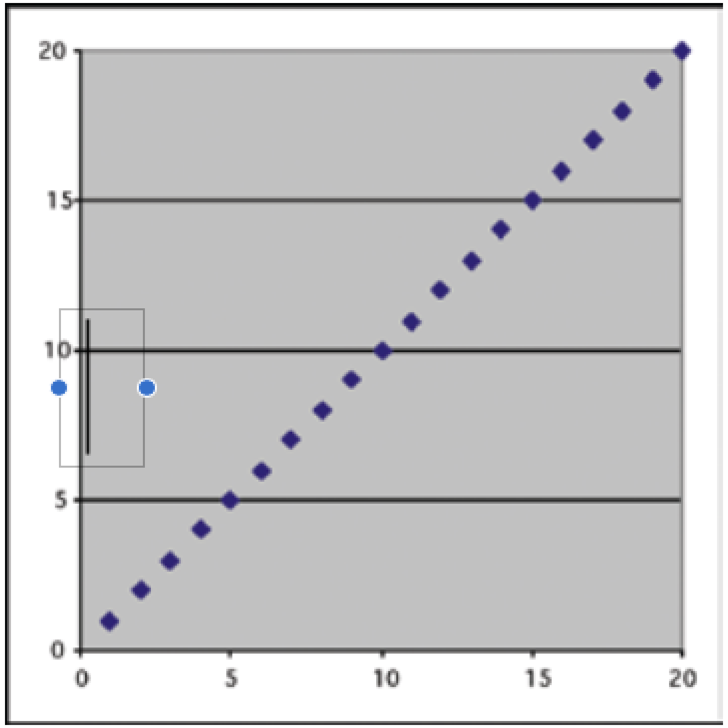
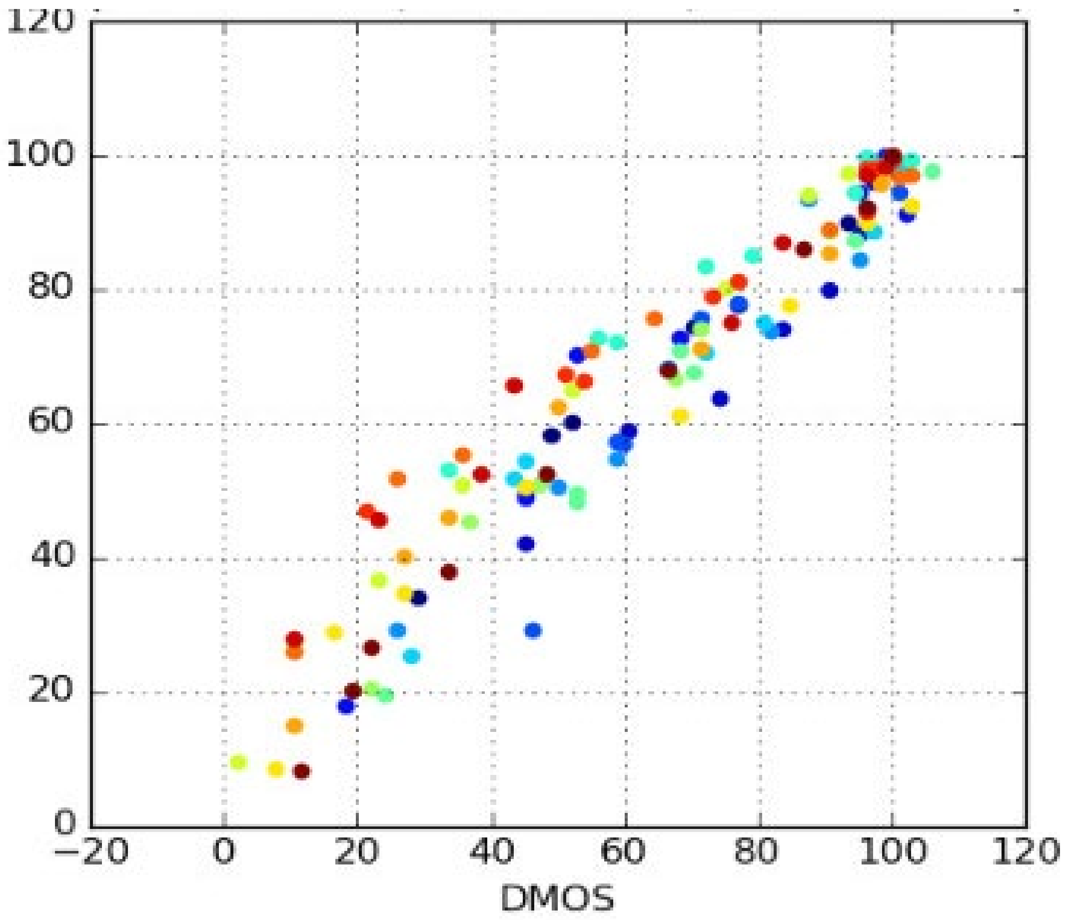


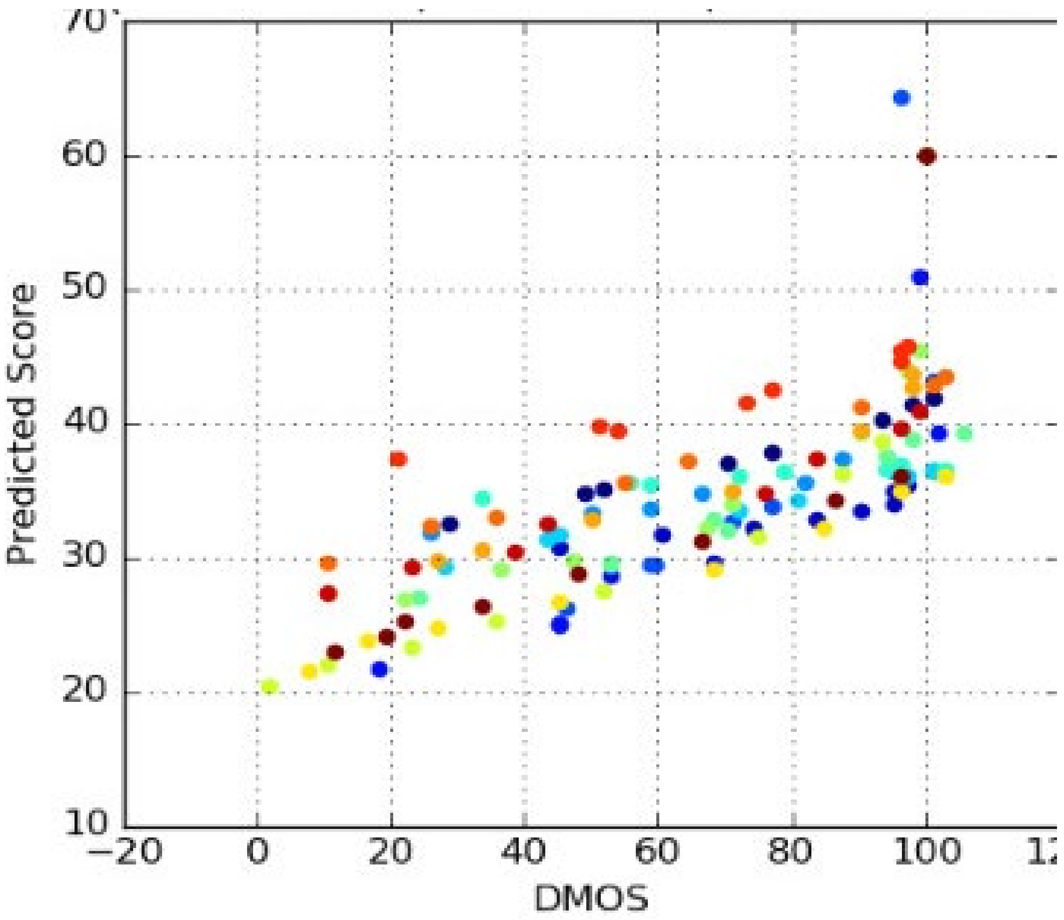
Figure 10. Correlation coefficient: 1.

Correlation with Subjective - VMAF

VMAF

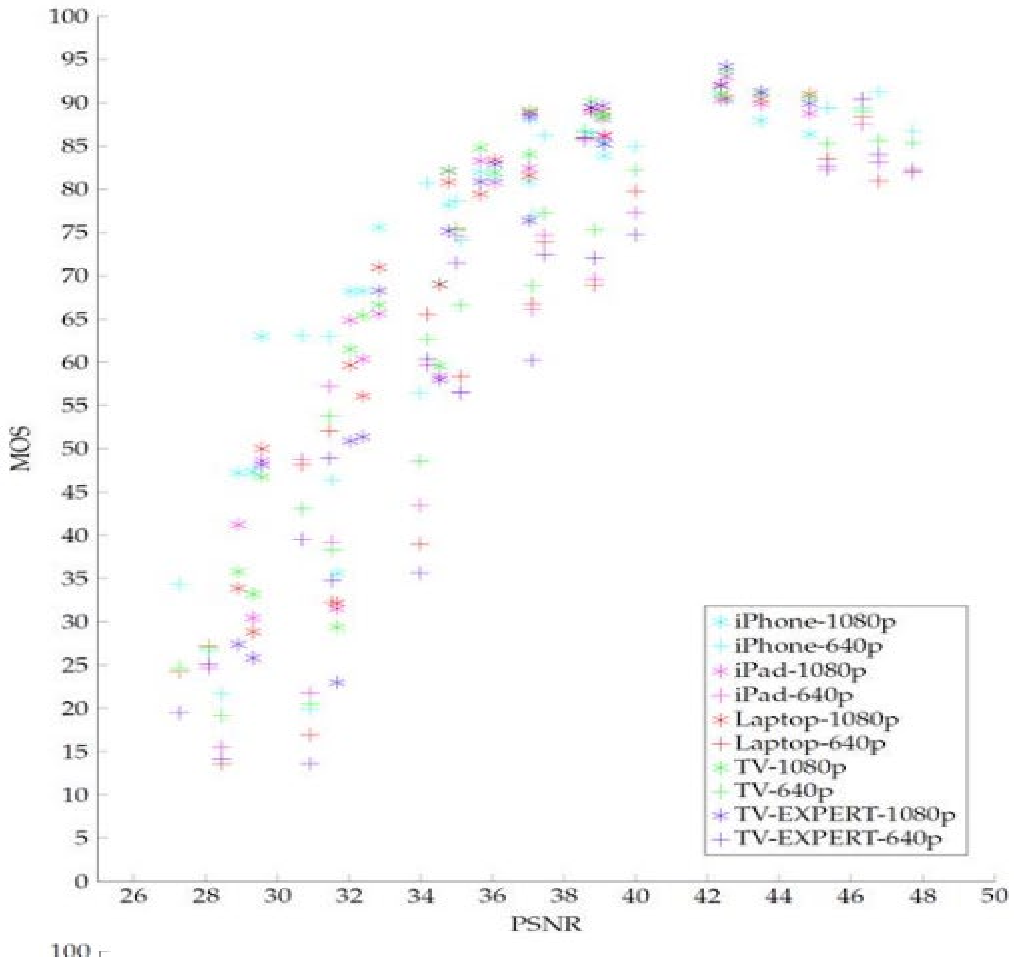


PSNR

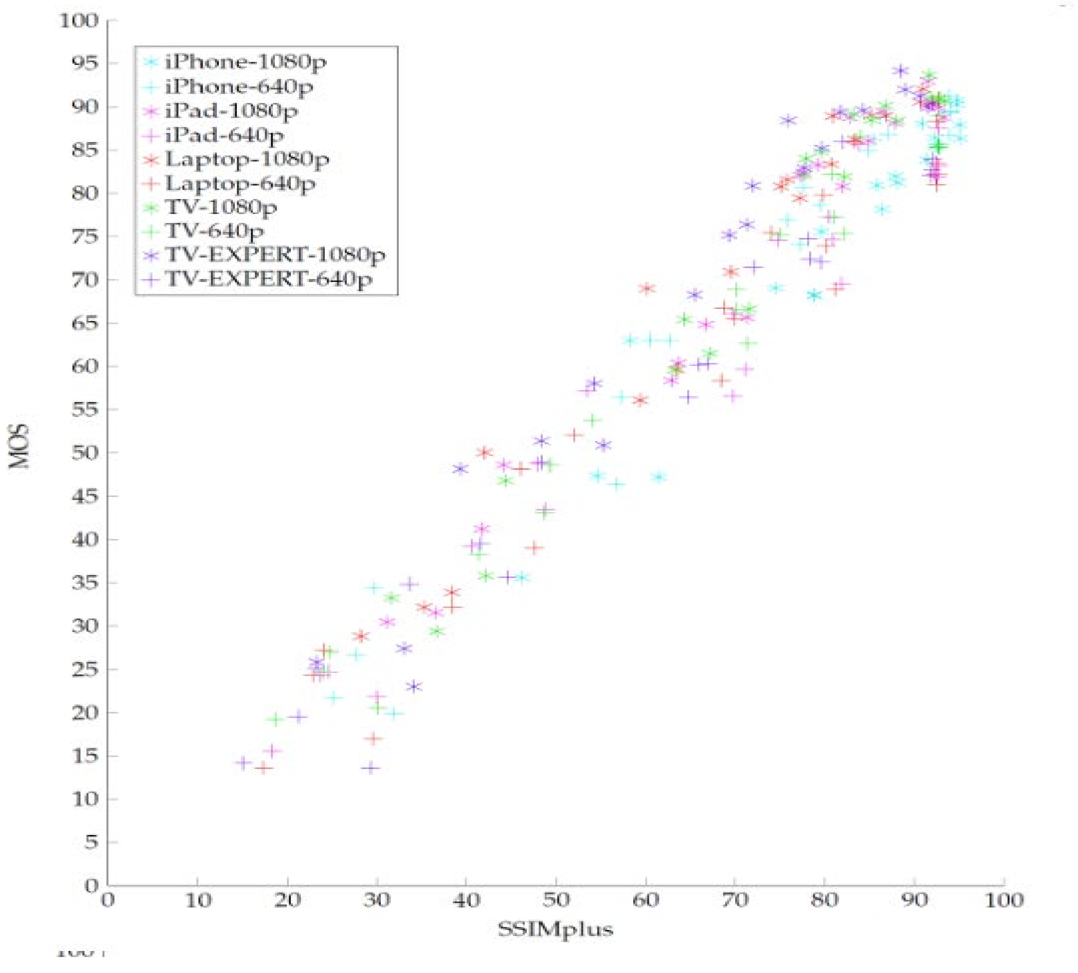


Correlation with Subjective - SSIMPLUS

PSNR

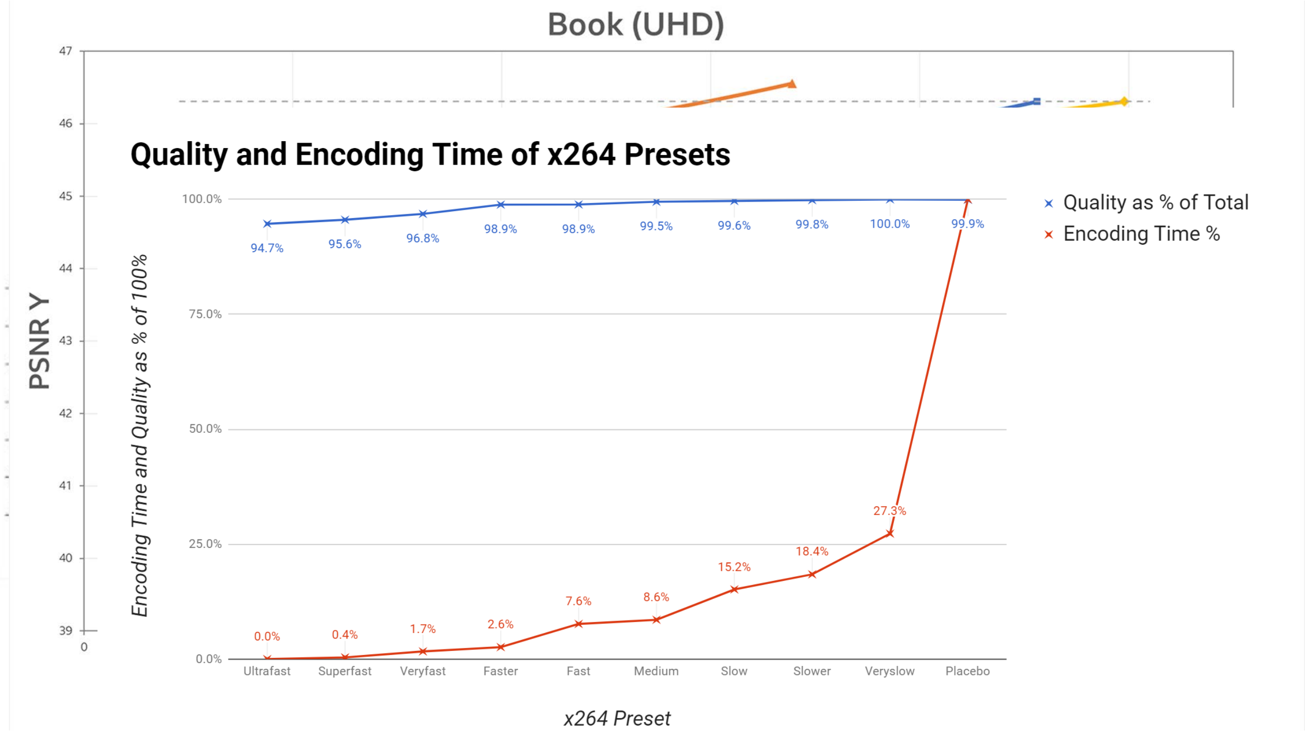


SSIMPLUS



How Are They Used

- Netflix
  - Per-title encoding
  - Choosing optimal data rate/rez combination
- Facebook
  - Comparing AV1, x265, and VP9
- Researchers
  - BBC comparing AV1, VVC, HEVC
- My practice
  - Compare codecs and encoders
  - Build encoding ladders
  - Make critical configuration decisions



Day to Day Uses

- Optimize encoding parameters for cost and quality
- Configure encoding ladder
- Compare codecs and encoders
- Evaluate per-title encoding technologies
- Add objectivity and rigor to any encoding-related decision

Alternatives for Subjective Comparisons

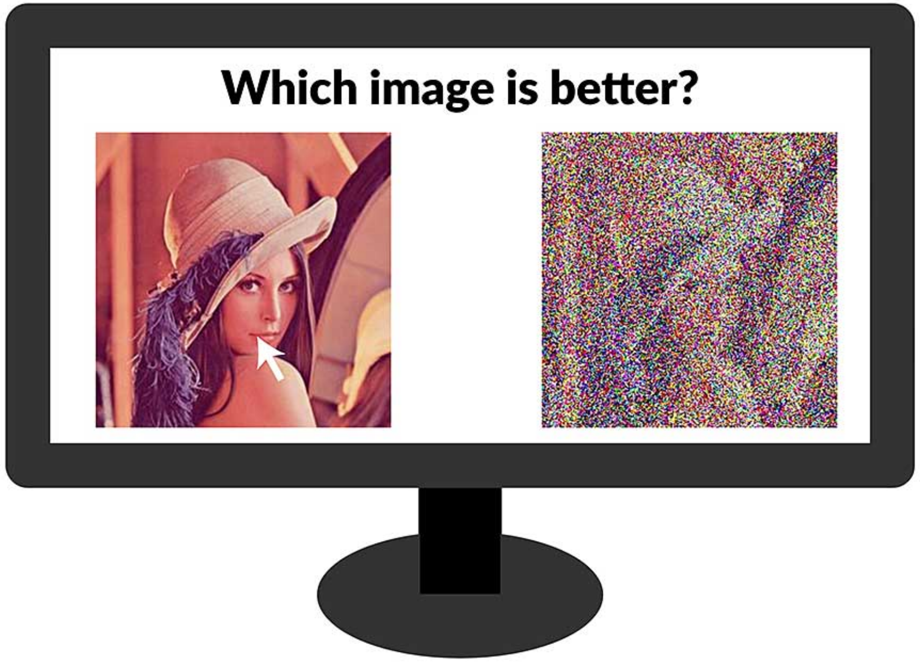
- Standards-based
  - ITU –R BT.500-13: Methodology for the subjective assessment of the quality of television pictures (bit.ly/ITU_R_BT500)
  - P.910 : Subjective video quality assessment methods for multimedia applications (www.itu.int/rec/T-REC-P.910/en)
- Golden-Eye
  - Small number of people with known ability to rate videos in repeatable ways that correspond with more general subjective test results
  - Used by many large production houses

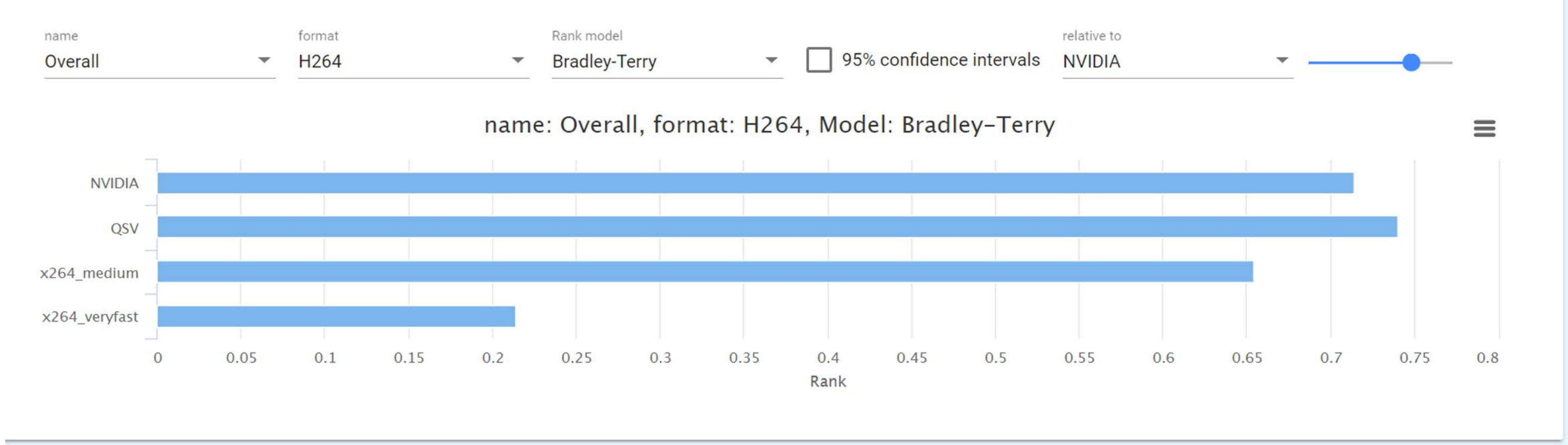
Subjective Evaluations

- What: Real viewers making real quality evaluations
- Pros
  - The "gold standard" measures actual human perception
- Cons
  - Slow and expensive
  - Shorter videos only due to attention spans

Alternatives for Subjective Comparisons

- Subjectify
  - A service from Moscow State University (bit.ly/Ozer_Subjectify)
  - Costs about $2/tester (for about 10 video comparisons each)
  - Used for multiple articles for Streaming Media and multiple consulting projects
  - Worth considering for important decisions





Questions

- Should be: 1:40

Lesson: Comparing Objective Metrics

- Overview
- Underlying mechanism
- Other features
  - Quality thresholds
  - Score correlation
  - Device ratings/models
  - Just noticeable difference (JDN)
  - SDR/HDR
  - Cross resolution
  - Cross frame rate
  - Cost/accessibility

Overview

- Goal: Make intelligent decisions
- Want metric that:
  - Has best correlation with subjective ratings
  - Provides relevant information
  - Provides actionable information

Underlying Mechanism (from a non-mathematician)

Mean Square Error

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

- Measures the cumulative squared error between the compressed and the original

Peak Signal to Noise

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

- Derivative of MSE, measures the ratio between the signal (true content) and the noise
- Both are math functions that don’t consider human visual functions
- Limits utility because humans don’t perceive all errors the same!

Underlying Mechanism (from a non-mathematician)

Structured Similarity Index (SSIM)

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

with:

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

- Perception-based model
- Incorporates luminance and contrast masking to compute *perceived* change
- Not just the difference between original and compressed, but how humans perceive the difference

Video Multimethod Assessment Fusion (VMAF)

- Combines 4 metrics
  - Visual Information Fidelity (VIF)
  - Detail Loss Metric (DLM)
  - Mean Co-Located Pixel Difference (MCPD)
  - Anti-noise signal-to-noise ratio (AN-SNR)
- Plus, machine learning
  - So, compute VMAF score
  - Perform subjective comparisons
  - Feed subjective results back into the VMAF formula to make the algorithm "smarter"
- Uses
  - Train for different types of content (animation, sports)
  - Train for different viewing conditions

Underlying Mechanism (from a non-mathematician)

SSIMPLUS

- Proprietary algorithm from the developer of SSIM
- Considers:
  - Temporal elements
  - Psycho-visual factors of human visual system
  - No machine learning but rapidly evolving

Accuracy Continuum

Less accurate → More accurate

Pure Math	Math + Perceptual	Math + Perceptual + Machine Learning
Mean Square Error	SSIM	SSIMPLUS
PSNR		VMAF

Other Considerations

- So, the most important factor is the ability to accurately predict subjective ratings
- Other factors
  - Quality thresholds
  - Score correlation
  - Just noticeable difference (JDN)
  - Device ratings/models
  - SDR/HDR
  - Cross resolution
  - Cross frame rate
  - Cost/accessibility

Other Factors: Quality Thresholds

- Quality thresholds
  - Does the metrics give you targets to shoot for?

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: Score Corrolation

- Quality thresholds
- Score correlation
  - Do scores correspond with subjective ratings
  - This simplifies interpreting scores and score differentials

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: Just Noticeable Difference

- Quality thresholds
- Score correlation
- Just noticeable difference (JDN)
  - Do you know what score differential should be noticeable?
  - When are scoring differences noticeable?

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: Device ratings/models

- Quality thresholds
- Score correlation
- Just noticeable difference (JDN)
- Device ratings/models
  - One score for all playback platforms?
    - From smartphone to 4K TV?
  - Different scores for different classes?
  - Different scores for different devices?

	PSNR	SSIM	SSIMplus	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: High Dynamic Range Ratings

- Quality thresholds
- Score correlation
- Just noticeable difference (JDN)
- Device ratings/models
- SDR/HDR
  - Grade HDR formatted videos

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: Cross Resolution/Cross Frame Rate

- Quality thresholds
- Score correlation
- Just noticeable difference (JDN)
- Device ratings/models
- SDR/HDR
- Cross resolution
- Cross frame rate
  - Can metric compute these or do you have to pre-convert encoded and/or source files
  - More a convenience factor

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Other Factors: Cost/Accessibility

- Quality thresholds
- Score correlation
- Just noticeable difference (JDN)
- Device ratings/models
- SDR/HDR
- Cross resolution
- Cross frame rate
- Cost/accessibility
  - Open-source metrics are often available for free in open-source tools
  - Proprietary metrics are typically available only in expensive tools and services.

	PSNR	SSIM	SSIMPLUS	VMAF
Scoring	0 – 100	0 – 1	0 - 100	0 - 100
No artifact threshold	45 dB	0.99	100	93
Artifacts likely present	35 dB	0.5	NA	NA
Interpreting scores				
Excellent	45+	.99 +	80 – 100	80 – 100
Good	38	.95 - .99	60 – 80	60 – 80
Fair	30	.88 - .98	40 – 60	40 – 60
Poor	24	.50-.88	20 – 40	20 – 40
Bad	< 15	< .5	< 20	< 20
Just Noticeable Difference	NA	NA	NA	6
Device ratings	No	No	Multiple TVs, monitors, devices	Standard, Phone, 4K
Grade HDR formats	No	No	Yes	No
Cross-resolution	No - convert	No - convert	Yes	No - convert
Cross-frame rate	No - convert	No - convert	Yes	No - convert
Cost/accessibility	Open source	Open source	Proprietary	Open source

Questions

- Should be: 1:50

Meet Video Multimethod Assessment Fusion

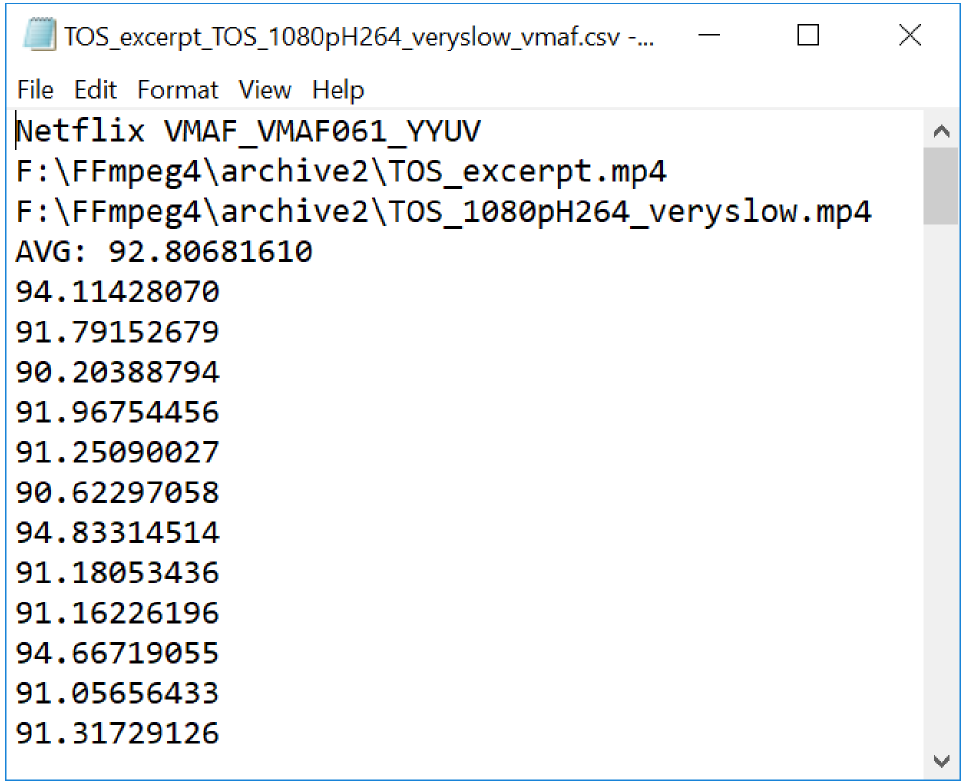
- What it is
- How accurate is it
- How to compute
- How to interpret
- Tools for computing

What is Video Multimethod Assessment Fusion

- Invented by Netflix
- Consolidation of four metrics (from Wikipedia)
  - Visual Information Fidelity (VIF): considers fidelity loss at four different spatial scales
  - Detail Loss Metric (DLM): measures detail loss and impairments which distract viewer attention
  - Mean Co-Located Pixel Difference (MCPD): measures *temporal* difference between frames on the luminance component
  - Anti-noise signal-to-noise ratio (AN-SNR)

What is VMAF?

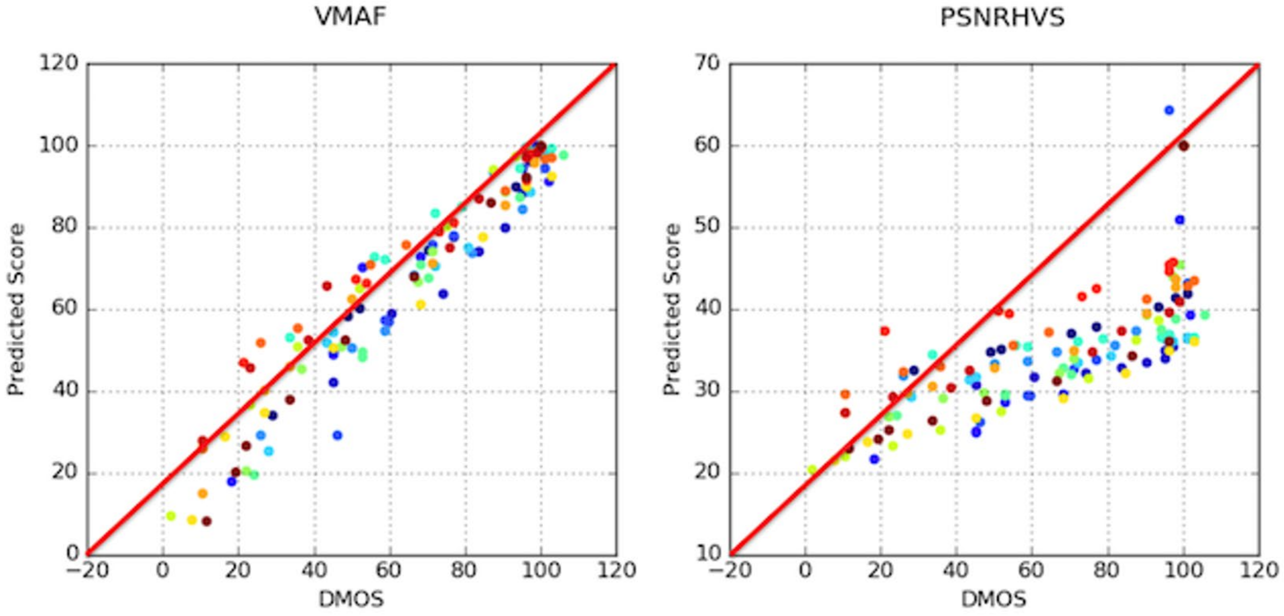
- Metrics are fused using a Support Vector Machine (SVM)-based regression to a single output score ranging from 0–100 per video frame
  - 100 being identical to the reference video
  - Frame values are averaged to compute a single score
  - So, a high score can mask many ugly frames (more later)



What is VMAF?

- VMAF is “trainable”
  - Compute VMAF
  - Measure human subjective ratings
  - Feed those results back into VMAF to make the algorithm “smarter”
- Uses
  - Train for different types of content (animation, sports)
  - Train for different viewing conditions

VMAF is a Good Predictor of Subjective Ratings



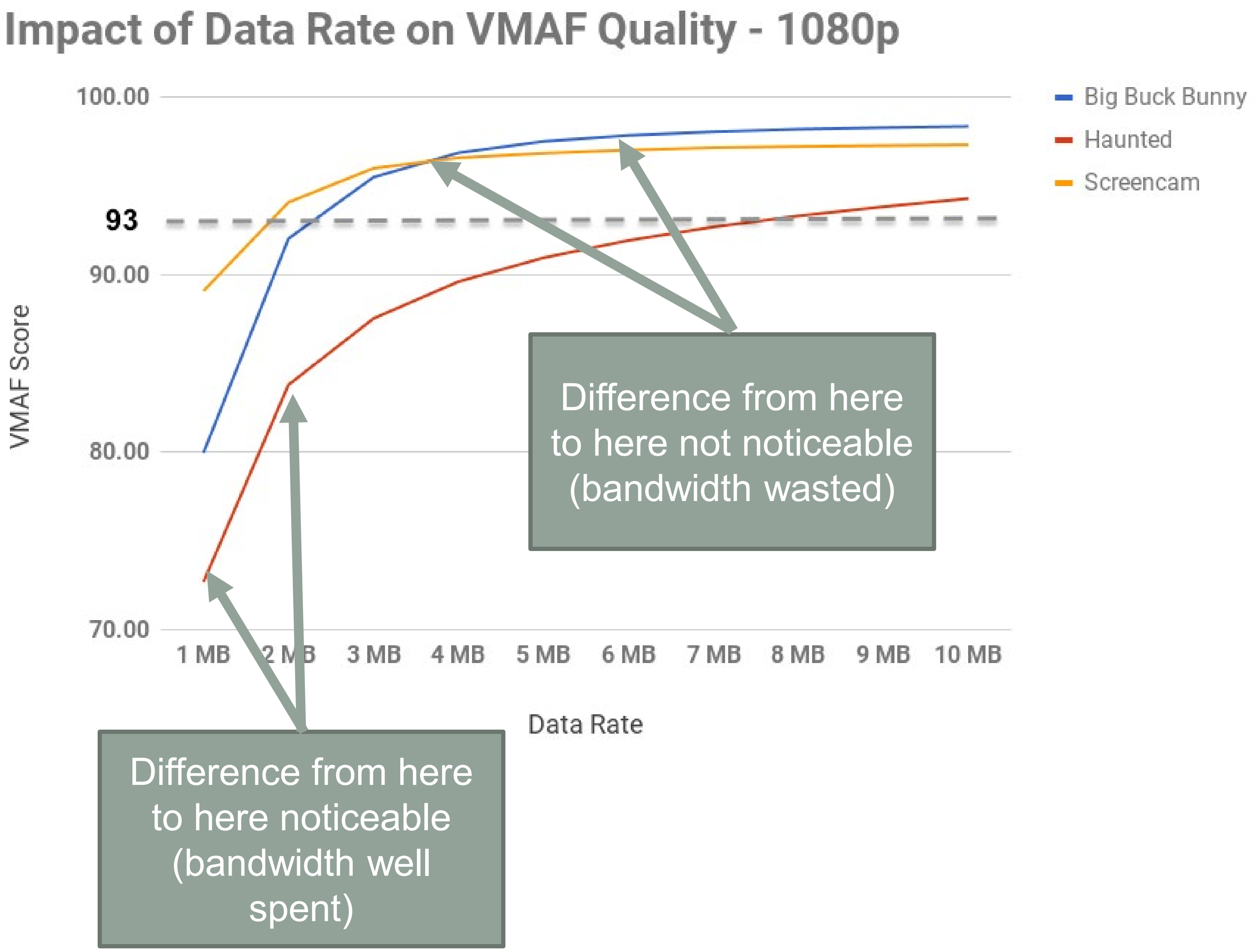
- Horizonal axis is DMOS rating (human scores)
- Vertical is metric (VMAF on left, PSNR on right)
- Red line is perfect score – metric exactly matches subjective evaluation
- VMAF is more tightly clumped around red line, which means it's more accurate
  - Machine learning means it can get more accurate over time
- PSRN is much more scattered, and as a fixed algorithm, will never improve

Working with VMAF – 93 is the Number

- Real Networks White Paper - VMAF Reproducibility: Validating a Perceptual Practical Video Quality Metric
  - 4K 2D videos
- ***VMAF score of about 93** … **is either indistinguishable from original or with noticeable but not annoying distortion.***
  - http://bit.ly/vrqm_5

Working With VMAF

- Scores map to subjective
  - 0-20 bad
  - 20 – 40 poor
  - 40 – 60 fair
  - 60 – 80 good
  - 80 – 100 excellent
- 6 VMAF points = Just noticeable difference



VMAF Models

- Original (Default) model
  - Assumed that viewers watch a 1080p display with the viewing distance of 3x the screen height (3H).
- Phone model
  - Assume viewers watch on a mobile phone
- 4K Model
  - Video displayed on a 4K TV and viewed from a distance of 1.5H

1080p display

Mobile Phone

4K display

VMAF Strengths

- Designed by Netflix specifically for use in multi-resolution comparisons
  - Comparing multiple resolutions at same data rate to ID highest quality (green background)
  - From my perspective, best metric for analyzing rungs on encoding ladder
- Trainable metric
- Living metric – Netflix/others continue to improve

H.264	1080p	720p	540p	432p	360p	270p	234p
5000	96.22						
4800	96.01						
4600	95.80	95.27					
4400	95.55	95.10					
4200	95.30	94.96					
4000	94.96	94.73					
3800	94.60	94.53					
3600	94.14	94.30					
3400	93.70	93.99					
3200	93.11	93.64					
3000	92.48	93.24					
2800	91.70	92.78					
2600	90.75	92.25					
2400	89.70	91.59	90.39				
2200	88.37	90.80	89.76				
2000	86.72	89.85	88.95	86.93			
1800	84.68	88.66	88.00	86.10			
1600	82.13	87.13	86.77	85.02	81.58		
1400	78.65	85.19	85.16	83.67	80.28		
1200	73.91	82.56	83.01	81.84	78.57		
1000	67.39	78.86	80.02	79.24	76.19		
900	63.18	76.39	77.98	77.47	74.60	66.66	60.58
800	57.93	73.25	75.51	75.34	72.68	65.11	59.23
700	51.47	69.42	72.34	72.59	70.23	63.14	57.49
600	43.12	64.52	68.37	69.11	67.12	60.70	55.33
500	33.31	58.05	63.13	64.66	63.04	57.52	52.46
400	20.82	49.48	56.00	58.46	57.48	53.13	48.59
300	9.74	37.56	45.95	49.62	49.60	46.80	42.96
200	3.73	20.40	30.87	36.12	37.48	36.88	34.03
100		2.75	8.08	14.45	17.50	19.85	18.66

VMAF Weaknesses

- No cross-resolution support
  - Must scale manually in most tools
  - Later lessons will cover
- No cross-frame rate support
  - Must create source file at encoded frame rate
- No support high dynamic range

Computing VMAF

- Moscow State University VQMT - $995 (covered later)
- Hybrik Cloud – at least $1,000/month (covered later)
- VMAF Master – Free (covered later)
- FFmpeg (covered later)
- Elecard Video Quality Estimator - $850

Questions

- Should be: 2:00

Meet SSIMPLUS

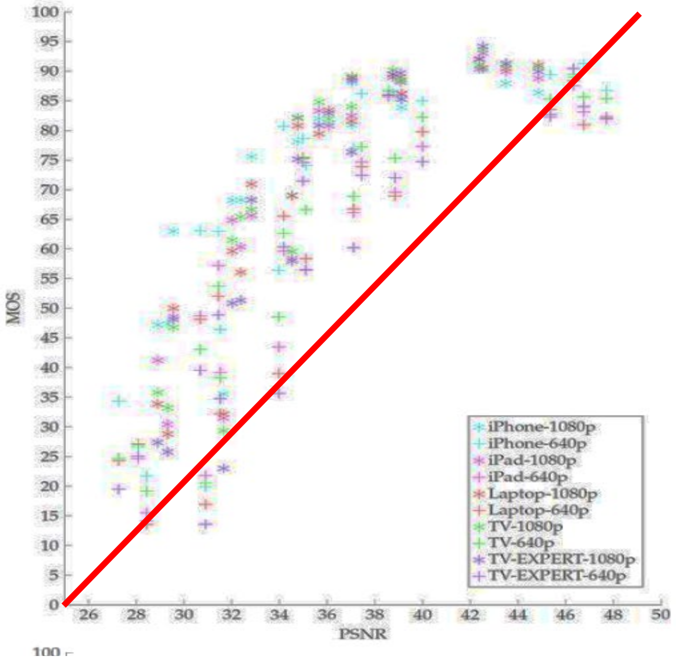
- What it is
- How accurate is it
- How to interpret scores
- Strengths and weaknesses
- Tools for computing

What is SSIMPLUS?

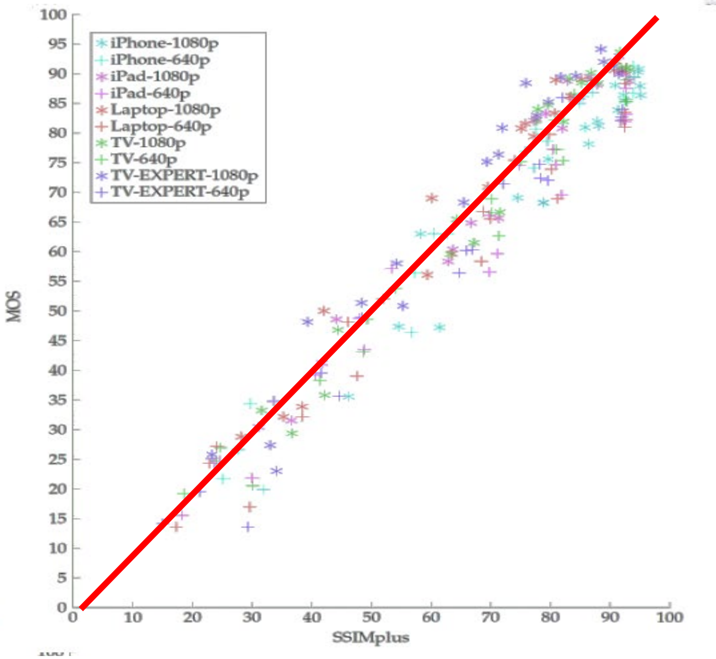
- Based on SSIM, extended to target video applications
- Strong correlation with subjective evaluations
- Scores map to easily understandable subjective ratings
- Supports multiple resolutions
- Supports multiple frame rates
- Supports some HDR formats
- Includes multiple device profiles
- Very fast

SSIMPLUS is a Very Good Predictor of Subjective Ratings

PSNR



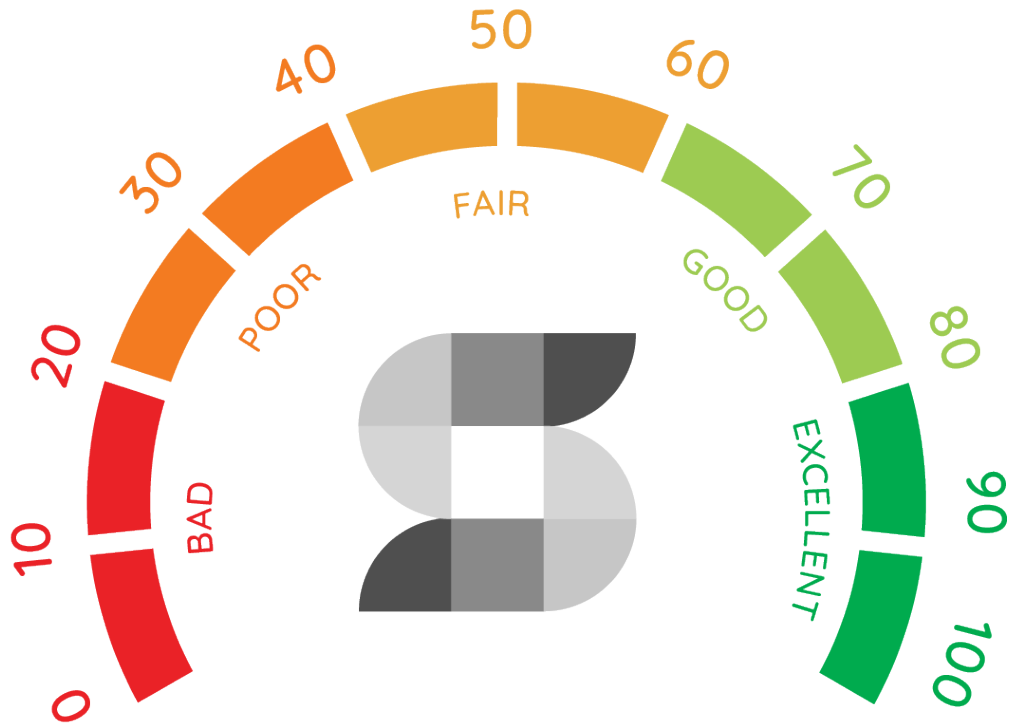
SSIMPLUS



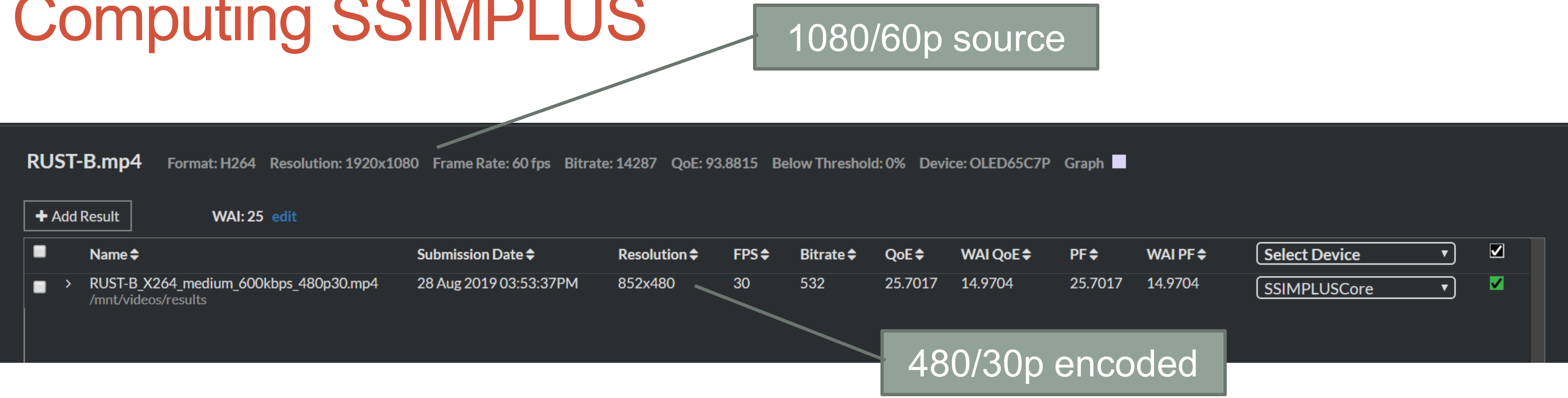
- Vertical axis is MOS rating (human scores)
- Horizontal is the metric (PSNR on left, SSIMPLUS on the right)
- Red line is perfect score, where the metric exactly matches subjective evaluation
- SSIMPLUS is more tightly centered around red line, which means it's more accurate
- PSRN is much more scattered
- SSIMWAVE claims an over 90% correlation with subjective ratings

Working With SSIMPLUS

- SSIMPLUS scores easily map to subjective ratings
  - 0-20 bad
  - 20 – 40 poor
  - 40 – 60 fair
  - 60 – 80 good
  - 80 – 100 excellent



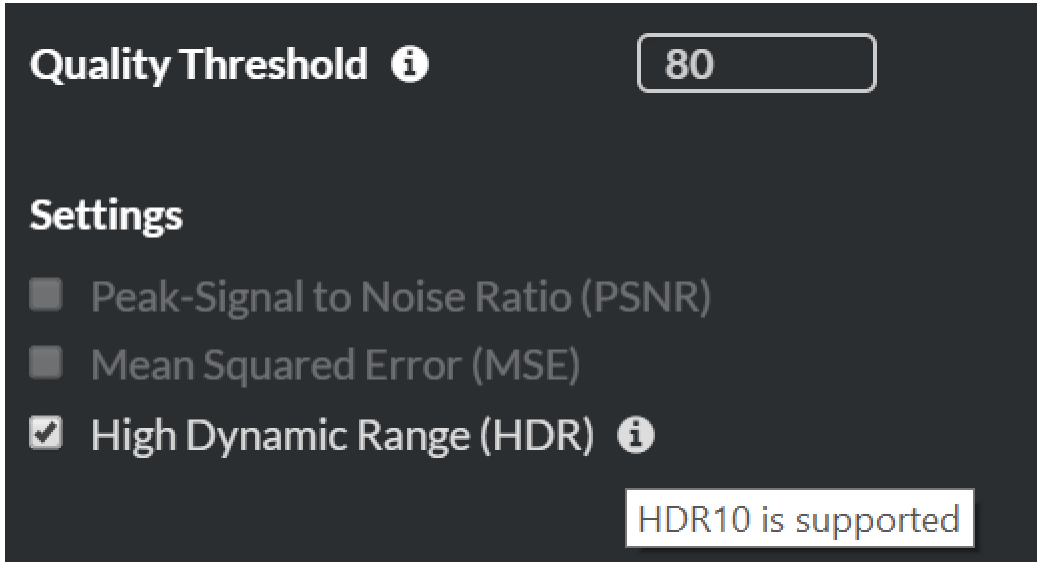
Computing SSIMPLUS



- SSIMPLUS can compare files with different resolutions than their source files
  - So no pre-scaling is necessary
- SSIMPLUS can compare files with different frame rates than source
  - No frame rate conversions required
  - SSIMPLUS is the only tool that can factor interframe smoothness into the frame rate comparisons

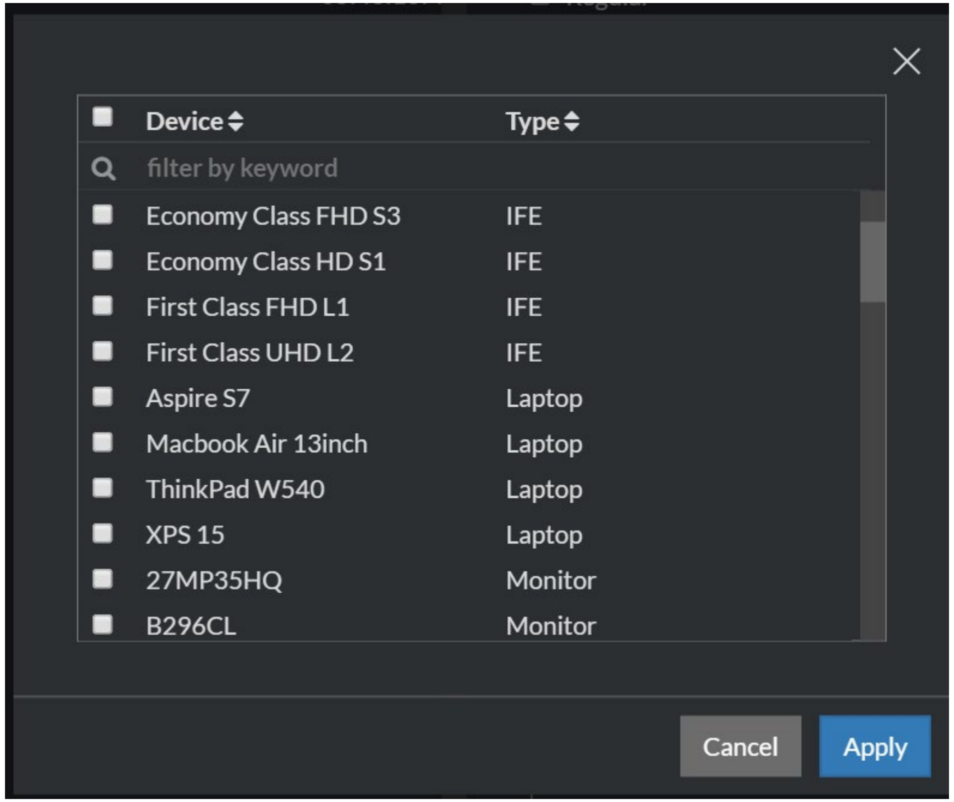
Currently Supports HDR10

- Only metric to incorporate HDR
- HDR10 supported with additional formats to come



SSIMPLUS Device Models

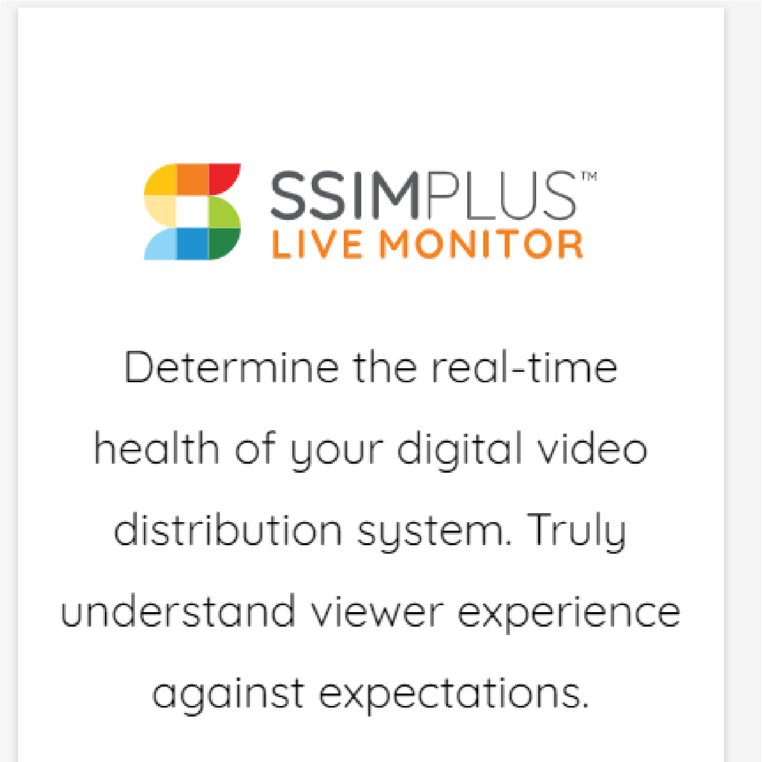
- All scores reported for generic device plus unlimited number of specific devices
  - Airline LCD panels
  - Smartphones
  - Tablets
  - Computer monitors
  - 1080p and 4K television sets
- Can assess quality on any and all devices relevant to your business
- Can customize encoding ladders by device



SSIMPLUS Performance

- A real-time or faster algorithm
- Available for both VOD and Live

SSIMPLUS Compatible Tools



- Both from SSIMWAVE
- SSIMPLUS VOD Monitor
  - Covered in this tutorial
- SSIMPLUS Live Monitor
  - Not addressed

SSIMPLUS Weaknesses

- Proprietary algorithm so only available on tools from inventor SSIMWAVE
- No concept of a Just Noticeable Difference
  - Unlike VMAF where 6 points is a JND
- The algorithm isn't trainable by users; all advances must come from SSIMWAVE

Questions

- Should be: 2:10

Meet PSNR

- What it is
- How accurate is it
- How to compute
- How to interpret
- Tools for computing

What is PSNR

- Static mathematical computation
  - No learning
- Used for still images
  - No concept of motion
  - Average frame values to compute score

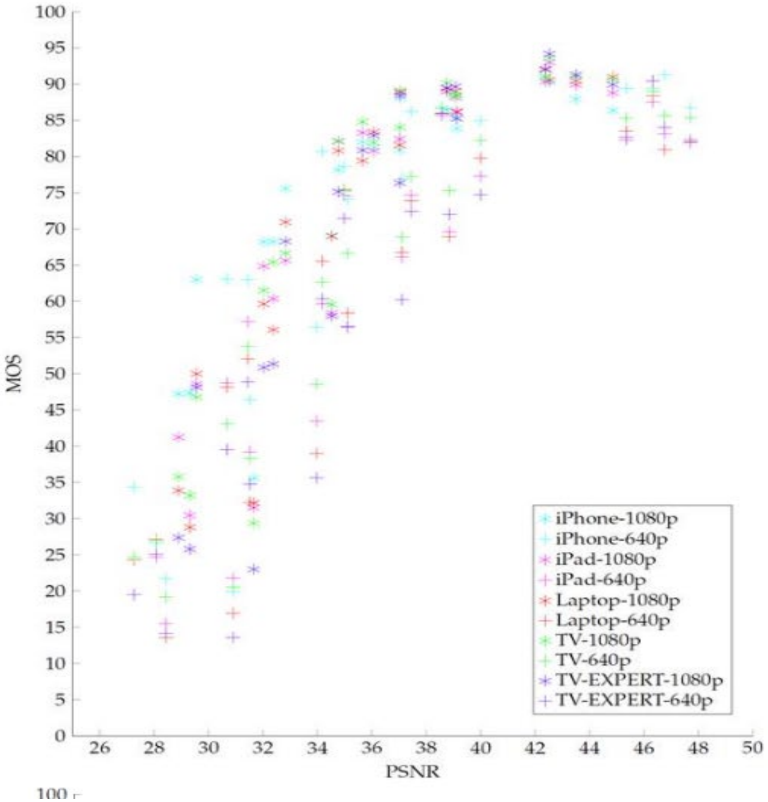
$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

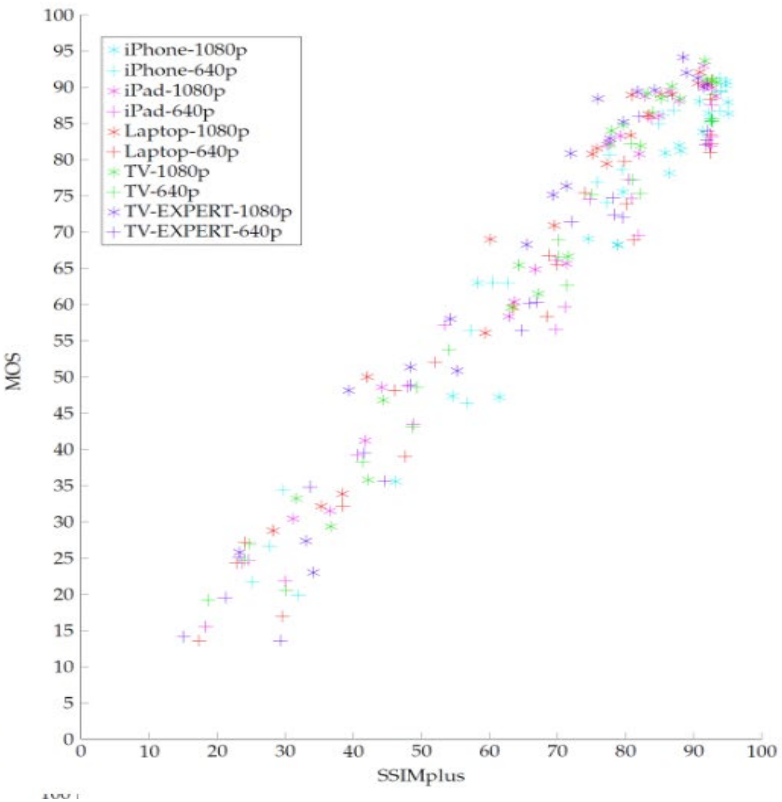
The PSNR (in dB) is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

How Accurate is PSNR?





- Loudly decried as inaccurate when announcing other metrics
  - Netflix and VMAF
  - SSIMWAVE and SSIMPLUS
- Still very widely cited because best known
  - Netflix, Facebook
  - Most academic/analytical studies

Accuracy Continuum

Less accurate → More accurate

Pure Math	Math + Perceptual	Math + Perceptual + Machine Learning
Mean Square Error	SSIM	SSIMPLUS
PSNR		VMAF

Computing PSNR – Same as VMAF

Source

4K Source

Encode

4K output

2K output

1080p output

720p output

480p output

Compare to:

4K Source

Which PSNR?

```
[libx264 @ 0000024f0cc1e900] ref B L0: 84.1% 12.3%  3.7%
[libx264 @ 0000024f0cc1e900] ref B L1: 93.9%  6.1%
[libx264 @ 0000024f0cc1e900] PSNR Mean Y:43.278 U:45.615 V:45.718 Avg:43.911 Global:43.593 kb/s:2508.55
[aac @ 0000024f0c820980] Qavg: 1011.472

F:\OQM>
```

- Many tools provide multiple outputs (Mean Y, Mean U, Mean V, Average, Global)
  - Y is luma (black and white/detail)
  - U/V are color
- Most report/use Mean Y (43.278)

How to Interpret PSNR

Range – 0 – 100 (in Decibels)

	PSNR
Scoring	0 – 100
No artifact threshold	45 dB
Artifacts likely present	35 dB
Interpreting scores	
Excellent	45+
Good	38
Fair	30
Poor	24
Bad	< 15
Just Noticeable Difference	NA

Higher than 45 delivers no perceivable quality improvement

Expect artifacts at 35 dB and lower

Correlations to subjective

No concept of JND

PSNR Strengths

- Familiarity
- Easy to access
- Does OK with same-resolution comparisons

PSNR Weaknesses

- No machine learning – will never improve
- No HDR
- No cross-resolution (scale in FFmpeg)
- No cross-frame rate (create comparable source in FFmpeg)

PSNR Bottom Line

- Developed as a still image metric; no concept of motion
- Used primarily for “reference” when producing metrics to share with the world
- Acceptable performance in same resolution testing (1080p to 1080p)
- Limited value (IMHO) when comparing files with different resolutions
- My use
  - Include in articles for reference; particularly codec/encoder comps
  - Included in consulting projects for reference
  - For books and other works moving to VMAF

Computing PSNR

- Moscow State University VQMT - $995 (covered later)
- Hybrik Cloud – at least $1,000/month (covered later)
- VMAF Master – Free (covered later)
- FFmpeg (covered later)
- Elecard Video Quality Estimator - $850

Questions

- Should be: 2:20

Meet Structural Similarity Index (SSIM)

- (Time Permitting)
- What it is
- How accurate is it
- How to compute
- How to interpret
- Tools for computing

What is SSIM

- Static mathematical computation
  - Incorporates some human perceptual modeling
  - No learning
- Designed for still images and video

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

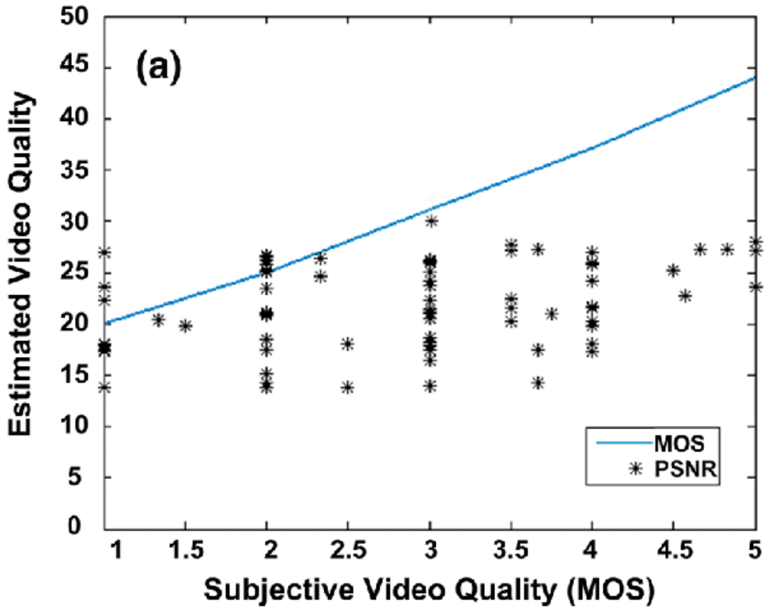
with:

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

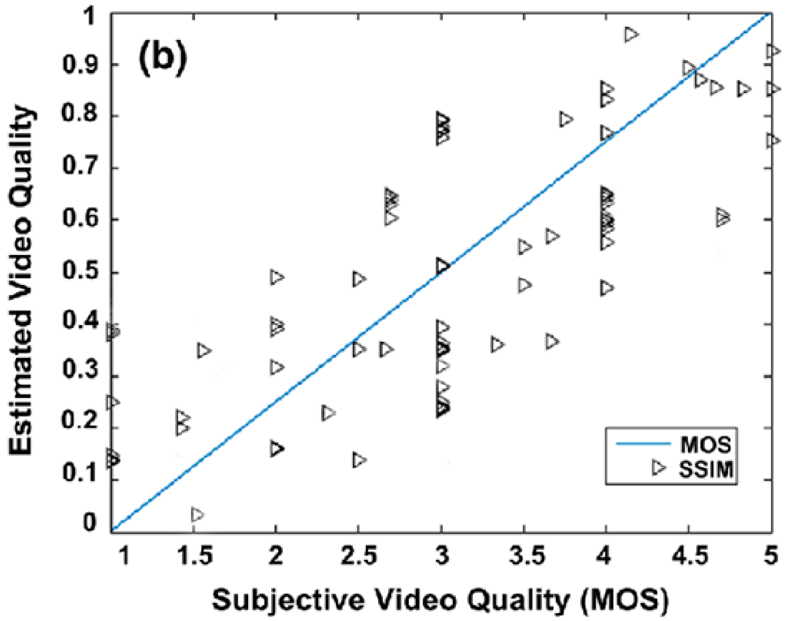
https://en.wikipedia.org/wiki/Structural_similarity

How Accurate is SSIM?

PSNR



SSIM



bit.ly/SSIMv_PSNR

- As shown on right, more accurate then PSNR
- This is the general perception of SSIM

Accuracy Continuum

Less accurate → More accurate

Pure Math	Math + Perceptual	Math + Perceptual + Machine Learning
Mean Square Error PSNR	SSIM	SSIMPLUS VMAF

Computing SSIM – Same as PSNR/VMAF

Source

4K Source

Encode

4K output

2K output

1080p output

720p output

480p output

Compare to:

4K Source

How to Interpret SSIM

Range – 0 – 1
Major issue (for me). Just not enough range between scores.

Higher than .99 delivers no perceivable quality improvement

Expect artifacts at .5 and lower

Correlations to subjective

No concept of JND

	SSIM
Scoring	0 – 1
No artifact threshold	0.99
Artifacts likely present	0.5
Interpreting scores	
Excellent	.99 +
Good	.95 - .99
Fair	.88 - .98
Poor	.50-.88
Bad	< .5
Just Noticeable Difference	NA

SSIM Strengths

- Familiarity
- Easy to access
- Higher accuracy rate than PSNR

SSIM Weaknesses

- No machine learning – will never improve
- Very small range 0 - 1
- No HDR
- No cross-resolution (scale in FFmpeg)
- No cross-frame rate (create comparable source in FFmpeg)

SSIM Bottom Line

- Used primarily for “reference” when producing metrics to share with the world
- Acceptable performance in same resolution testing
- Limited value (IMHO) when comparing files with different resolutions

Computing SSIM

- Moscow State University VQMT - $995 (covered later)
- Hybrik Cloud – at least $1,000/month (covered later)
- VMAF Master – Free (covered later)
- FFmpeg (covered later)
- Elecard Video Quality Estimator - $850

Questions

- Should be: 2:20

Computing Metrics

- Lesson: Workflows
- Lesson: FFmpeg
- Lesson: VMAF Master
- Lesson: Moscow State University Video Quality Measurement Tool
- Lesson: SSIMWAVE VOD Inspector

Lesson: Metric Workflows

- Reference vs. non-reference metrics
- How reference metrics work
- Working with lower resolution files
- Working with different frame rates
- Tuning for metrics

Reference vs. Non-Reference

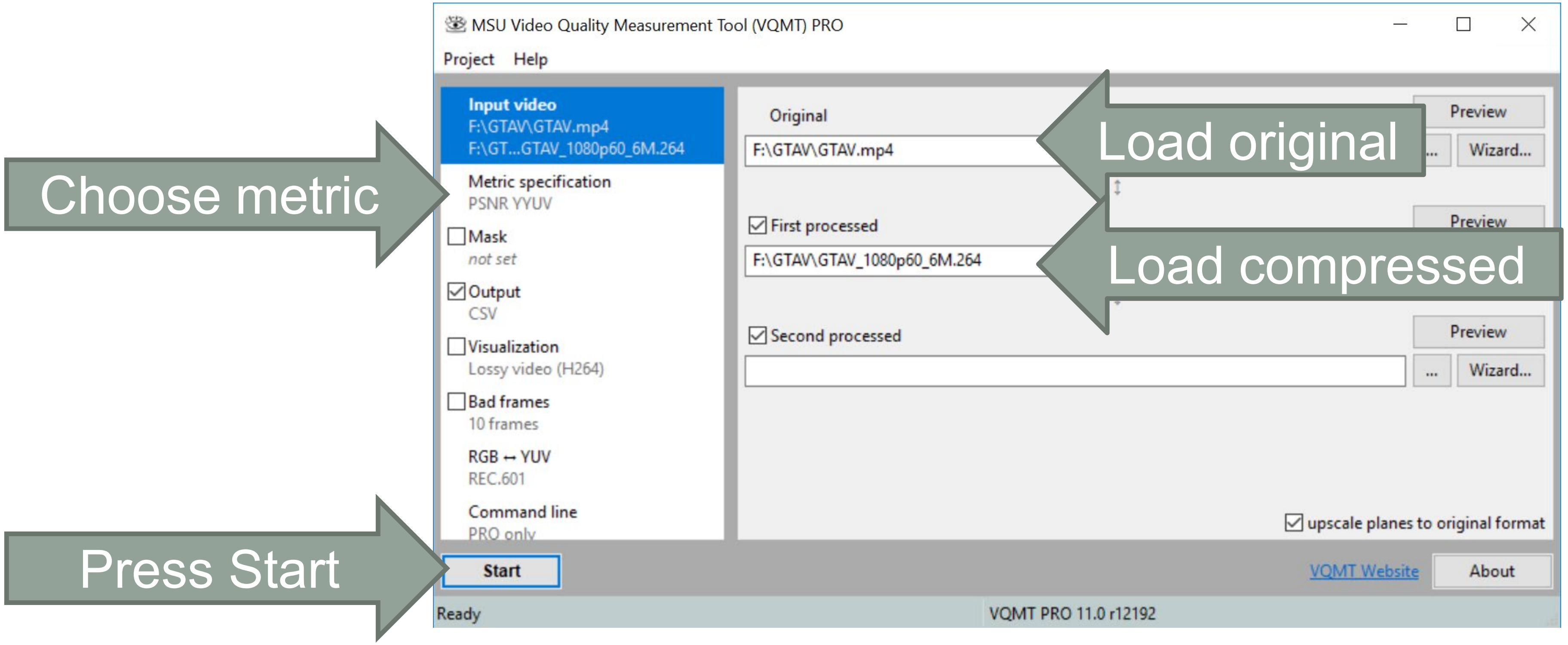
Reference

- Compare the encoded file to the original
  - Need original file to compute
  - Can't compute "downstream" in distribution pipeline
- Generally considered the most accurate
- Very difficult to produce in real time
  - So not useful for live

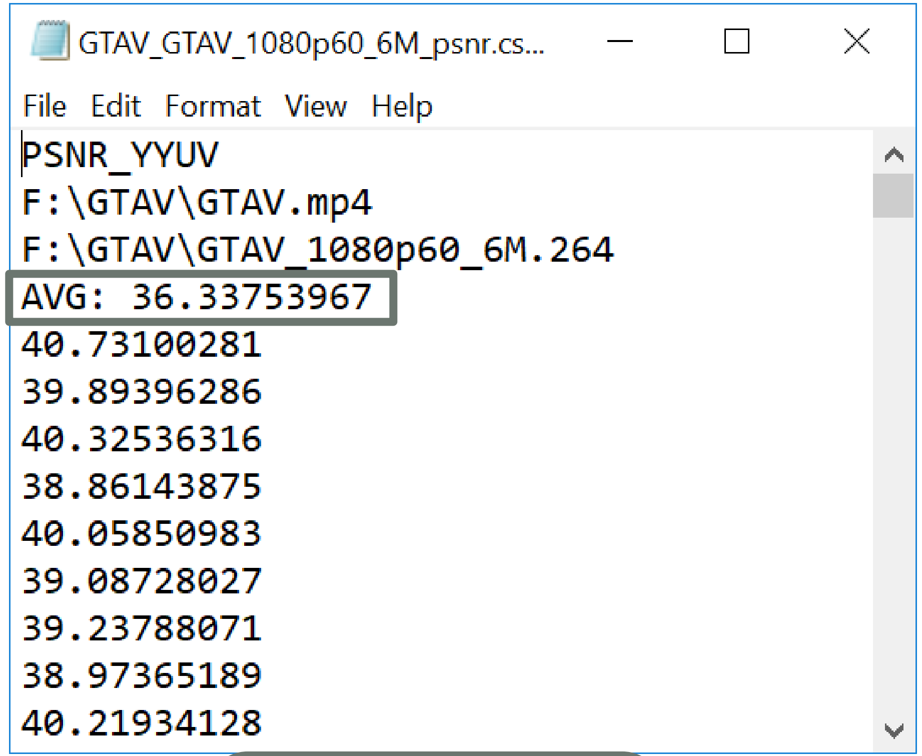
Non-Reference

- Analyzes only the compressed file; doesn't need original
- Generally considered less accurate than referential but getting better
- Can be real time/live
- Can analyze files downstream in the distribution pipeline

How Reference Metrics Work



Tool Computes the Metric: Delivers the Score



Some show visualization (MSU VQMT)

All provide the score

Working with Lower Resolution Files

Source

4K Source

Encode

4K output

2K output

1080p output

720p output

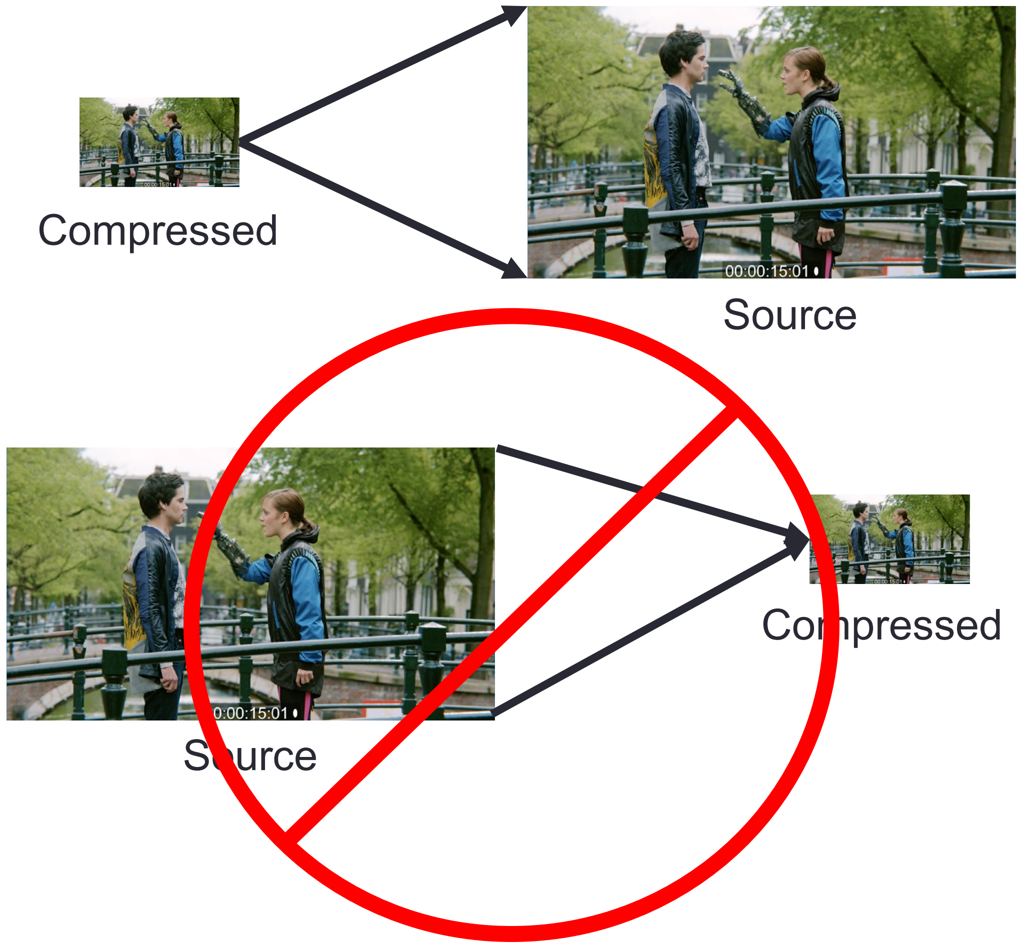
480p output

Compare to:

4K Source

Working with Lower Resolution Files

- Most metrics can only compare files of equal resolution
- So, you scale ***compressed*** videos to ***source*** rez
- Either manually beforehand
  - Usually with FFmpeg (covered in a different lesson)
- Or metric tool scales behind the scenes
- You ***don't*** scale source rez to encoded rez



Scaling Low Resolution Files to Source Resolution

- Why?
  - Because most metrics only compare files of like resolution
- Exceptions?
  - Some metrics/tools will scale for you in the background (SSIMWAVE, VQMT version 11.1 +)
  - For most others (FFmpeg, VMAF Master) you must scale beforehand
- How
  - In FFmpeg

Working with Different Frame Rates

60 fps

30 fps

- Most encoding ladders for 60 fps footage have 30 fps streams
- Most metrics can only compare footage with same frame rate
- For most (not all) tools, you have to create a 30 fps source file using FFmpeg
- This measures frame quality, but not the smoothness component

Questions

- Should be: 2:30

Tuning for Metrics

- What is it?
- Why?
- How?
- Who is doing what?
- General rules?

What is Tuning?

- Disable features that:
  - Improve subjective video quality but
  - Degrade objective scores
- Example: adaptive quantization – changes bit allocation over frame depending upon complexity
  - Improves visual quality
  - Looks like “error” to metrics like PSNR/VMAF

What is Tuning?

- Switches in encoding string that enables tuning (and disables these features)

```
ffmpeg -input.mp4 -c:v libx264 -tune psnr output.mp4
```

- With x264, this disables adaptive quantization and psychovisual optimizations

Why So Important

- Major point of contention:
  - "If you're running a test with x264 or x265, and you wish to publish PSNR or SSIM scores, you MUST use –tune PSNR or –tune SSIM, or your results will be completely invalid."
  - http://x265.org/compare-video-encoders/
- Absolutely critical when comparing codecs because some may or may not enable these adjustments
- You don't have to tune in your tests; but you should address the issue and explain why you either did or didn't

Does Impact Scores

- 3 mbps football (high motion, lots of detail)
- PSNR
  - No tuning – 32.00 dB
  - Tuning – 32.58 dB
  - .58 dB
- VMAF
  - No tuning – 71.79
  - Tuning – 75.01
  - Difference – over 3 VMAF points
    - 6 is JND, so not a huge deal
    - But if inconsistent between test parameters, could incorrectly show one codec (or encoding configuration) as better than the other

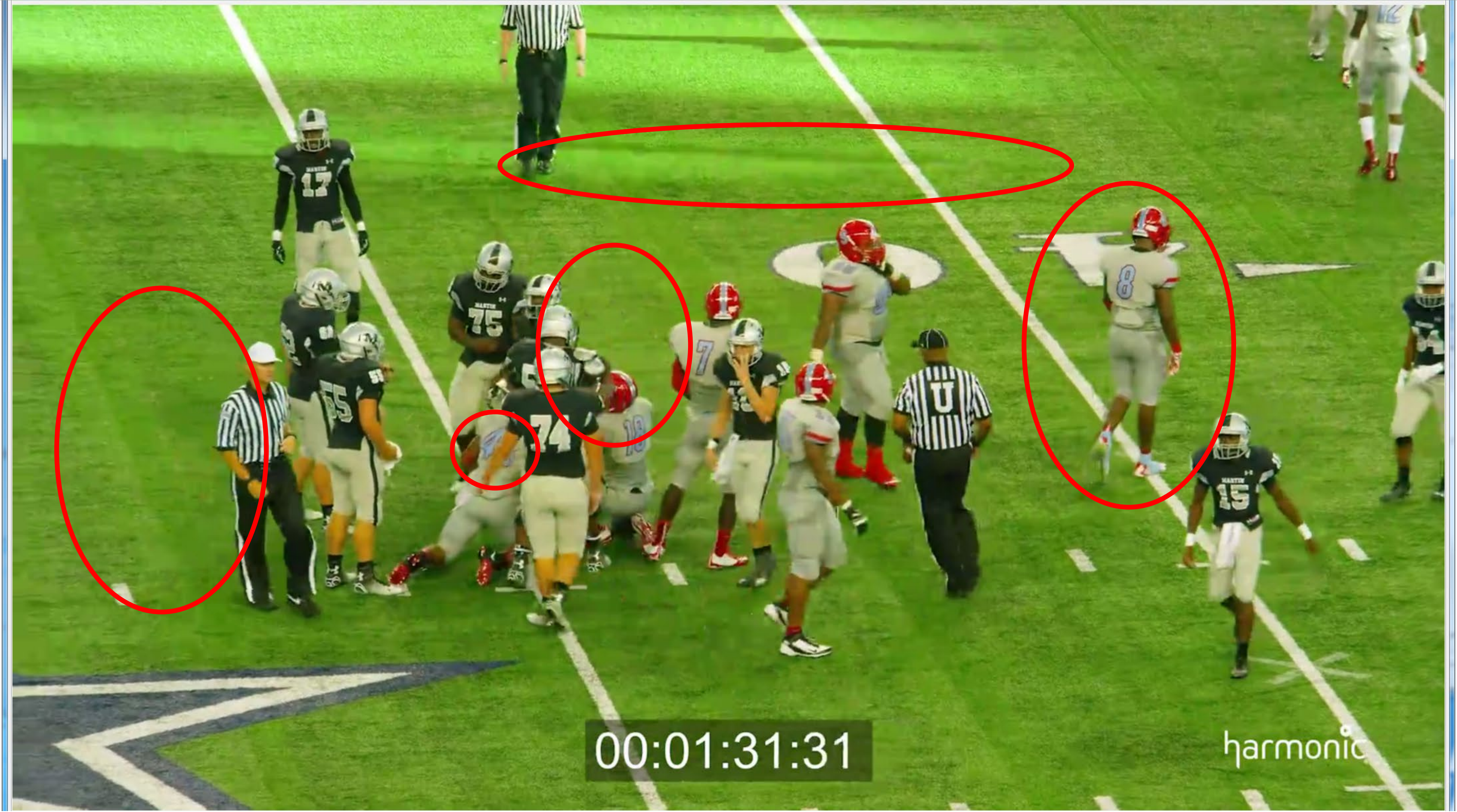
VQMT VMAF Graph

Red – tuned
Green – not tuned

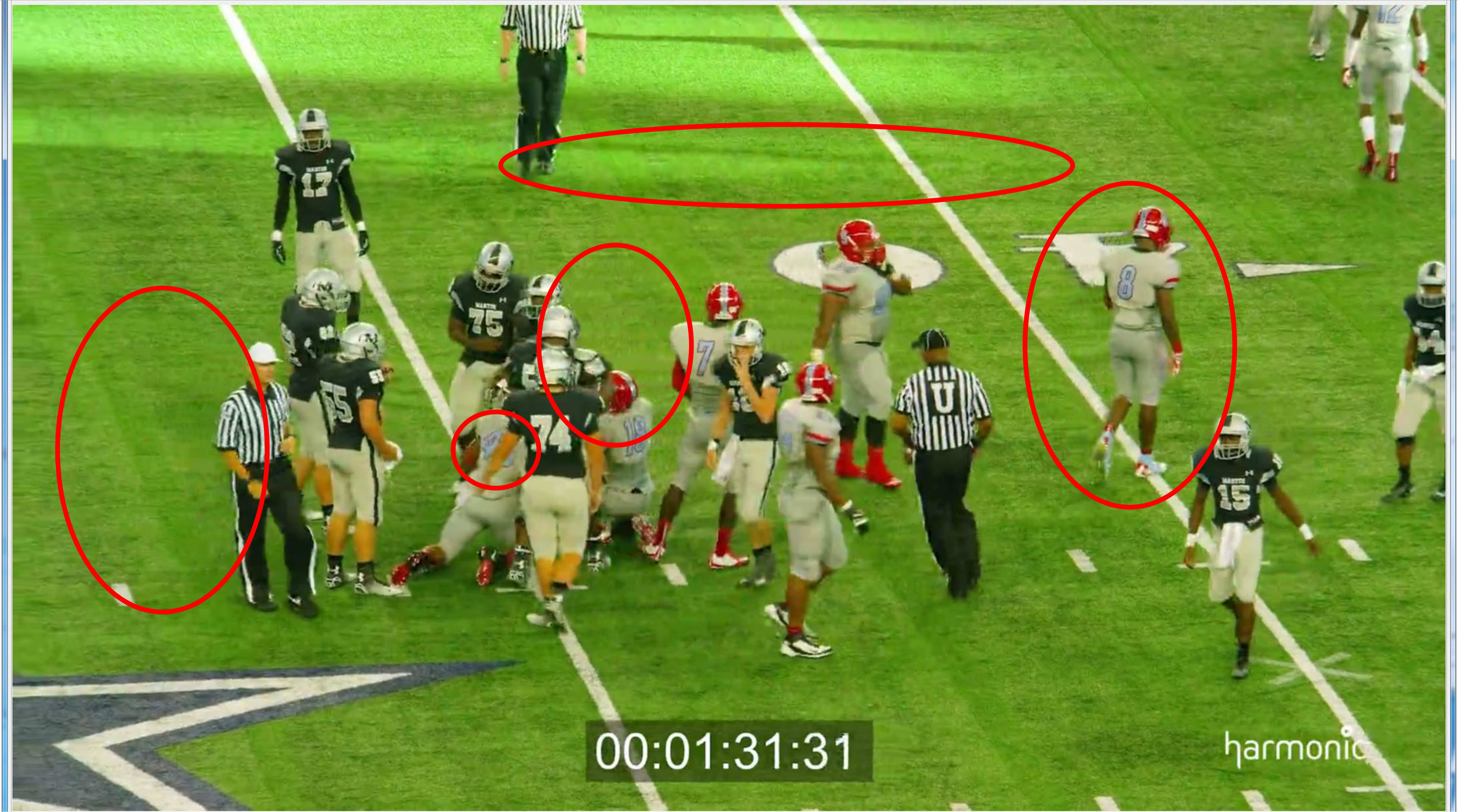
Multiple frames with
3-4-point differentials

Downward spikes represent untuned frames that metric perceives as having lower quality

Tuned



Not tuned



Observations

- Tuning
  - Produces more blurry areas
  - Reduces detail
  - Reduces artifacts
- Without tuning
  - More detail
  - Slightly more artifacts
  - Looks more accurate and “better” to my eye
- Key point:
  - When comparing encoders and codecs with visual quality metrics, **be consistent**
    - If tuning for one, tune for all
- When comparing encoding parameters with the same codec, not so critical
  - Tuning or not tuning should have the same efffect

Most Academic Comparisons Tend to Tune

- Coding efficiency comparison of AV1/VP9, H.265/MPEG-HEVC, and H.264/MPEG-AVC encoders
  - bit.ly/Grois_AV1

TABLE I. SELECTED SETTINGS FOR THE AOM/AV1 ENCODER

CODEC	AOM/AV1
Version	AOMedia Project AV1 Encoder, Version: b6724815f22876ca88f43b57dba09a555ef4e1b0
Recommended settings	--best --psnr --tune=psnr --end-usage=q --passes=2 --tile-columns=0 --arnr-strength=5 --min-q=$QP --max-q=$QP --cq-level=$QP

TABLE II. SELECTED SETTINGS FOR THE X265 ENCODER

CODEC	x265
Version	VideoLAN Project x265 Encoder, Version: 2.0
Recommended settings	--profile=main –p=placebo --psnr --tune=psnr --pools none --no-pmode --no-pme --no-allow-non-conformance --rd=6 --rect --amp –qp=$QP --keyint=$IntraPeriod --min-keyint=$IntraPeriod --pass=2

Moscow State University

- MSU Codec Comparison 2018
  - bit.ly/MSU_HEVC_18
  - Tuned whenever possible

Universal Encoding	x264 --preset slower --me hex --keyint infinite --tune ssim --pass 1 --bitrate %BITRATE_KBPS% %SOURCE_FILE% --input-res %WIDTH%x%HEIGHT% --fps %FPS% -o NUL x264 --preset slower --me hex --keyint infinite --tune ssim --pass 2 --bitrate %BITRATE_KBPS% %SOURCE_FILE% --input-res %WIDTH%x%HEIGHT% --fps %FPS% -o %TARGET_FILE%

Universal Encoding	x265.exe --input %SOURCE_FILE% --input-res %WIDTH%x%HEIGHT% --fps %FPS% -p medium --bitrate %BITRATE_KBPS% --psnr --ssim --tune=ssim -o %TARGET_FILE% --bframes 4 --max-merge 3 --ref 3 --b-intra --limit-ref 1 --early-skip

Practitioners Are Mixed

- Facebook
  - AV1 beats x264 and libvpx-vp9 in practical use cases
    - bit.ly/FB_AV1_VP9
  - Two encoding cases, ***neither tuned***

Codec	CRF/QP mode	ABR mode
AV1	<INPUT> --i420 -y --codec=av1 --cpu-used=1 --threads=0 --profile=0 --lag-in-frames=19 --min-q=0 --max-q=63 --auto-alt-ref=1 --kf-max-dist=60 --kf-min-dist=60 --drop-frame=0 --static-thresh=0 --bias-pct=50 --minsection-pct=0 --maxsection-pct=2000 --arnr-maxframes=7 --arnr-strength=5 --sharpness=0 --undershoot-pct=100 --overshoot-pct=100 --tile-columns=0 --frame-parallel=0 --test-decode=warn -v --end-usage=q --cq-level=<CRF> --webm -o <OUTPUT>	<INPUT> --i420 -y --codec=av1 --cpu-used=1 --threads=0 --profile=0 --lag-in-frames=19 --min-q=0 --max-q=63 --auto-alt-ref=1 --passes=<PASS> --kf-max-dist=60 --kf-min-dist=60 --drop-frame=0 --static-thresh=0 --bias-pct=50 --minsection-pct=0 --maxsection-pct=2000 --arnr-maxframes=7 --arnr-strength=5 --sharpness=0 --undershoot-pct=100 --overshoot-pct=100 --tile-columns=0 --frame-parallel=0 --test-decode=warn -v --end-usage=vbr --target-bitrate=<BITRATE> --webm -o <OUTPUT>
x264 Main Profile	-i <INPUT> -c:v libx264 -pix_fmt yuv420p -profile:v main -preset veryslow -crf <CRF> -refs 5 -g 60 -keyint_min 60 -sc_threshold 0 -f mp4 <OUTPUT>	-i <INPUT> -c:v libx264 -pix_fmt yuv420p -profile:v main -preset veryslow -b:v <BITRATE> -refs 5 -g 60 -keyint_min 60 -sc_threshold 0 -pass <PASS> -f mp4 <OUTPUT>
x264 High Profile	-i <INPUT> -c:v libx264 -pix_fmt yuv420p -profile:v high -preset veryslow -crf <CRF> -refs 5 -g 60 -keyint_min 60 -sc_threshold 0 -f mp4 <OUTPUT>	-i <INPUT> -c:v libx264 -pix_fmt yuv420p -profile:v high -preset veryslow -b:v <BITRATE> -refs 5 -g 60 -keyint_min 60 -sc_threshold 0 -pass <PASS> -f mp4 <OUTPUT>
libvpx-vp9	-i <INPUT> -c:v libvpx-vp9 -pix_fmt yuv420p -crf <CRF> -b:v 0 -speed 1 -tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -keyint_min 60 -g 60 -f webm <OUTPUT>	-i <INPUT> -c:v libvpx-vp9 -pix_fmt yuv420p -b:v <BITRATE> -speed 1 -tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -keyint_min 60 -g 60 -pass <PASS> -f webm <OUTPUT>

Practitioners Are Mixed

- Netflix – Doesn't Tune
  - Standardization bodies tend to use test conditions that let them compare one tool to another, often maximizing a particular objective metric and reducing variability over different experiments. For example, rate-control and visual tunings are generally disabled, to focus on the effectiveness of core coding tools.
  - Netflix encoding recipes focus on achieving the best quality, enabling the available encoder tools that boost visual appearance, and thus, giving less weight to indicators like speed or encoder footprint that are crucial in other applications.
  - bit.ly/NF_codecs

Netflix on Tuning

- Best Practices for Netflix's VMAF Metric
  - bit.ly/VMAF_bestp
- On tuning for VMAF
  - "Since VMAF partially captures the benefit of perceptual optimization, and if at the end of the day you will be encoding with these settings on, we still recommend turning them on."

General Rules

- When VQ metrics accurately mimic human perception, there will be no need to tune
- Until then:
  - Be consistent – either tune for all or don't tune for any
- If testing for publication, detail what you did and why
  - This decision will make or break public perception of your work
- If producing for inhouse use:
  - Test using actual production parameters unless this introduces an obvious bias

Implementing Tuning

- Tuning varies by codec
  - x264/x265 – can tune for PSNR/SSIM
  - Intel SVT-AV1 – can tune for PSNR/VMAF/Visual quality
  - NGCodec (others) – Must manually disable adaptive quantization
- Before getting started:
  - Check codec documentation
  - Spend an hour checking other published comparisons to see what they did

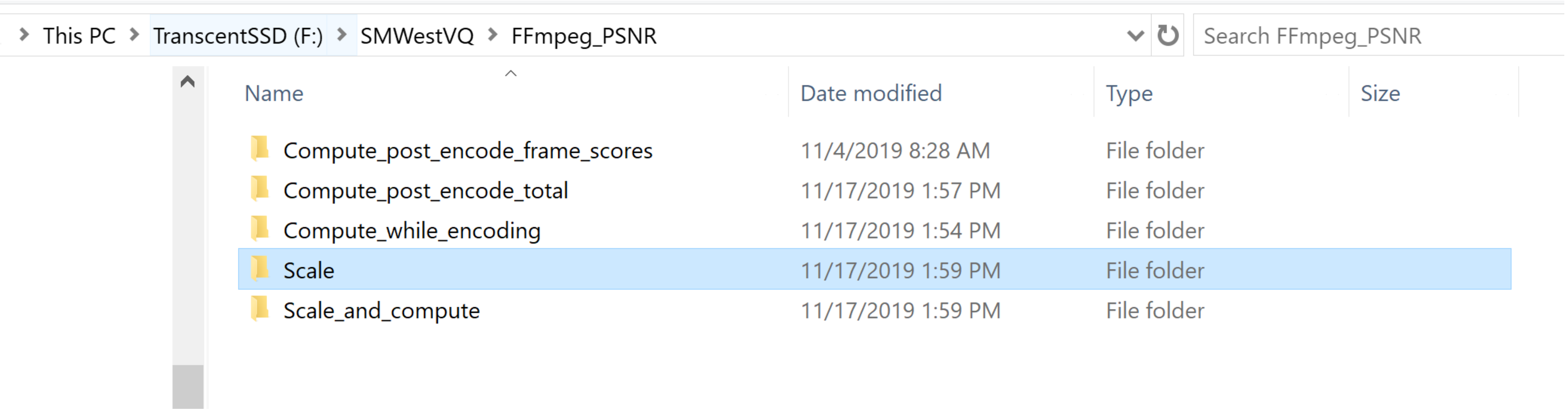
Questions

- Should be: 2:40

Computing PSNR with FFmpeg (Updated Session)

- Setup – part of standard FFmpeg installation
- File requirements
- While encoding
- Post-encode – average score
- Post-encode – average score/per-frame score

Folders


This PC
TranscentSSD (F:)
SMWestVQ
FFmpeg_PSNR
Search FFmpeg_PSNR
Name
Date modified
Type
Size
Compute_post_encode_frame_scores 11/4/2019 8:28 AM File folder
Compute_post_encode_total 11/17/2019 1:57 PM File folder
Compute_while_encoding 11/17/2019 1:54 PM File folder
Scale 11/17/2019 1:59 PM File folder
Scale_and_compute 11/17/2019 1:59 PM File folder

File Requirements (compute while encoding)

- Compute during encoding
  - Will compute PSNR if different rez/frame rate, but it will be incorrect
  - So, don't run if encoding 1080p file to 720p
- Post-encode
  - Resolution ***must*** be the same (scale before computing or create script that scales)

```
Press [q] to stop, [?] for help
[Parsed_psnr_0 @ 000002b57eb807c0] Width and height of input videos must be same.
[Parsed_psnr_0 @ 000002b57eb807c0] Failed to configure input pad on Parsed_psnr_0
Error reinitializing filters!
Failed to inject frame into filter network: Invalid argument
Error while processing the decoded data for stream #1:0
Conversion failed!
```

  - Frame rate ***should*** be the same
    - Unable to produce reliable results with 60 fps source and 30p output

Computing PSNR While Encoding

```
ffmpeg -i input.mp4 -c:v libx264  -tune psnr -b:v 3000K -report -psnr  output.mp4
```

- `-tune psnr` – since we're measuring PSNR we'll tune for PSNR
  - You'll see this error message if you don't tune

```
Press [q] to stop, [?] for help
[libx264 @ 000002b03653e040] --psnr used with psy on: results will be invalid!
[libx264 @ 000002b03653e040] --tune psnr should be used if attempting to benchmark psnr!
```

- `-report` – this produces a log file with the PSNR recorded; otherwise, you'll only be able to grab the score from the command window (see next slide)
- `-psnr` – tells FFmpeg to compute PSNR
- `Output.mp4` – output (compressed) file for post-encode computations
- Let's try

Results in Command Window

```
[libx264 @ 00000120e3930980] PSNR Mean Y:42.334 U:44.698 V:44.800 Avg:42.975 Global:42.680 kb/s:2487.53
[aac @ 00000120e1812ac0] Qavg: 172.987
```

- FFmpeg displays multiple outputs (Mean Y, Mean U, Mean V, Average, Global)
- Correct value is Mean Y (42.334)

Results in Log File

```
[libx264 @ 0000022c884d0980] PSNR Mean Y:42.334 U:44.698 V:44.800 Avg:42.975 Global:42.680 kb/s:2487.53
[aac @ 0000022c87c22a80] Qavg: 172.987
```

- FFmpeg will create a report named ffmpeg_date_time.log
- Scroll down to the bottom to see the same outputs as the Command window
- Most use Mean Y (42.334)

Scaling Low Resolution Files to Source Resolution

- Why?
  - Because most metrics only compare files of like resolution
- Exceptions?
  - Some metrics/tools will scale for you in the background (SSIMWAVE, VQMT version 11.1 +)
  - For most others (FFmpeg, VMAF Master) you must scale beforehand
- How
  - In FFmpeg

Scaling Low Resolution Files (scale)

```
ffmpeg -i input_720p.MP4 -pix_fmt yuv420p -vsync 0 -s 1920x1080 -sws_flags lanczos output_720p_2_1080p.y4m
```

- `-pix_fmt` – yuv420 works with all metrics tools. May need a higher quality format if HDR
- `-vsync 0` – maintains audio sync
- `-s 1920x1080` – set this to resolution of source video
- `-sws_flags lanczos` – this tells FFmpeg to use the Lanczos filter to scale. I use this because this is the filter NVIDIA uses in their graphics cards. Since we're trying to simulate graphics display quality it seemed to make sense. If you'd like to use a different filter (or leave it blank and use the default) that's fine, just be consistent.
- `output_720p_2_1080p.y4m` – Y4M files contain resolution, pixel format, and other metadata in the file header so you don't have to specify this in the command string or via the user interface. This makes Y4M easier to work with than YUV files in most instances. If you absolutely need a YUV file, change the file extension of the output file to output.yuv.
- Let's run it!

Copy y4m file to both compute post folders

Compute PSNR After Encoding – Total Only

```
ffmpeg -i input.mp4 -i output_720p_2_1080p.y4m -filter_complex "psnr" -report -f null -
```

- `Input.mp4` – source
- `output_720p_2_1080p.y4m` – encoded (copied from scale)
- `-filter_complex “psnr”` - calling this filter complex
- `-report` – to record scores in a log file; otherwise only appears in Command window
- `-f null -` – tells FFmpeg to output a null file (need –f null -)
- Let’s try

Compute post encode total folder

Compute PSNR After Encoding – Report File

```
[Parsed_psnr_0 @ 000002070ccaee80] PSNR y:40.221995 u:44.008545 v:43.913345 average:41.150038 min:39.378826
max:44.959762
[AVIOContext @ 000002070c1f4600] Statistics: 19990461 bytes read, 2 seeks
[AVIOContext @ 000002070c2be580] Statistics: 373248782 bytes read, 0 seeks
```

- FFmpeg will create a report named ffmpeg_date_time.log
- Scroll down to the bottom to see the same outputs as the Command window
- Most use Mean Y (40.221)

Scaling and Computing PSNR

```
ffmpeg -i input_720p.mp4 -i input.mp4 -filter_complex
[0v]scale=1920x1080:flags=lanczos[input_720p];[input_720p][1v]psnr -report -f
null -
```

- `[0v]scale=1920x1080:flags=lanczos[input_720p];` -scale first video (`[0v]`) to 1080p using lanczos method and label it `input_720p`
- `[input_720p][1v]psnr –` forward it to PSNR using the label `input_720p` and compare it to the first video `[1v]`
- `-report -f null –` as before

```
[Parsed_psnr_1 @ 000001b5a22c43c0] PSNR y:40.221995 u:44.008545 v:43.913345 average:41.150038 min:39.378826
max:44.959762
```

Scale and compute folder

Compute PSNR After Encoding – Frame Scores

```
ffmpeg -i output_720p_2_1080p.y4m -i input.mp4 -lavfi  psnr=output_3MB_psnr.log -report -f null
```

- `output_720p_2_1080p.y4m` – encoded (note reverse order from previous. copied from scale folder)
- `Input.mp4` – source
- `-lavfi  psnr=output_3MB_psnr.log` - calls Librafilter, computes psnr and inserts individual frame scores into this log file
  - Substitute desired name for name shown
  - Useful when you want to record individual frame scores
- `-report` – records overall scores; log file only records individual frame scores
- `-f null -` – tells FFmpeg to output a null file (need `-f null -`)
- Let's try

```
[Parsed_psnr_0 @ 00002562f8161c0] PSNR y:40.221995 u:44.008545 v:43.913345 average:41.150038 min:39.378826
max:44.959762
```

Compute post encode frame scores folder

Compute PSNR After Encoding – PSNR Log

```
n:1 mse_avg:1.47 mse_y:1.59 mse_u:1.38 mse_v:1.07 psnr_avg:46.47 psnr_y:46.12 psnr_u:46.73 psnr_v:47.85
n:2 mse_avg:2.36 mse_y:2.88 mse_u:1.50 mse_v:1.12 psnr_avg:44.41 psnr_y:43.54 psnr_u:46.38 psnr_v:47.63
n:3 mse_avg:2.30 mse_y:2.80 mse_u:1.49 mse_v:1.11 psnr_avg:44.52 psnr_y:43.66 psnr_u:46.40 psnr_v:47.69
n:4 mse_avg:2.77 mse_y:3.28 mse_u:2.01 mse_v:1.50 psnr_avg:43.70 psnr_y:42.97 psnr_u:45.10 psnr_v:46.36
n:5 mse_avg:2.27 mse_y:2.63 mse_u:1.78 mse_v:1.33 psnr_avg:44.57 psnr_y:43.93 psnr_u:45.63 psnr_v:46.88
n:6 mse_avg:2.59 mse_y:3.11 mse_u:1.78 mse_v:1.34 psnr_avg:44.00 psnr_y:43.21 psnr_u:45.62 psnr_v:46.87
n:7 mse_avg:2.98 mse_y:3.50 mse_u:2.21 mse_v:1.68 psnr_avg:43.38 psnr_y:42.69 psnr_u:44.68 psnr_v:45.87
n:8 mse_avg:2.85 mse_y:3.43 mse_u:1.96 mse_v:1.46 psnr_avg:43.57 psnr_y:42.78 psnr_u:45.21 psnr_v:46.50
n:9 mse_avg:2.50 mse_y:2.91 mse_u:1.93 mse_v:1.43 psnr_avg:44.15 psnr_y:43.50 psnr_u:45.27 psnr_v:46.57
n:10 mse_avg:3.25 mse_y:3.83 mse_u:2.38 mse_v:1.79 psnr_avg:43.02 psnr_y:42.30 psnr_u:44.37 psnr_v:45.60
n:11 mse_avg:2.79 mse_y:3.30 mse_u:2.02 mse_v:1.49 psnr_avg:43.68 psnr_y:42.94 psnr_u:45.08 psnr_v:46.41
n:12 mse_avg:2.99 mse_y:3.55 mse_u:2.12 mse_v:1.60 psnr_avg:43.38 psnr_y:42.63 psnr_u:44.87 psnr_v:46.09
```

- PSNR log contains individual frame scores
- Can input into Excel/Sheets for additional presentation or analysis

Consistency

- May be slight differentials between scores computed *while* encoding and *post* encoding
- Use same technique for all
  - If can’t can’t compute all during encode, compute post encode

Questions

- Should be: 2:50

Computing SSIM with FFmpeg (only if on Time)

- Setup – part of standard FFmpeg installation
- File requirements
  - While encoding – not available
  - Post-encode – average score
  - Post-encode – average score/per-frame score

File Requirements

- Post-encode
  - Resolution ***must*** be the same (scale before computing)

```
[Parsed_ssim_0 @ 000002799c28e140] Width and height of input videos must be same.
[Parsed_ssim_0 @ 000002799c28e140] Failed to configure input pad on Parsed_ssim_0
Error reinitializing filters!
Failed to inject frame into filter network: Invalid argument
Error while processing the decoded data for stream #1:0
Conversion failed!
```

  - Frame rate ***should*** be the same
    - Unable to produce reliable results with 30 fps source

Compute SSIM After Encoding – Total Only

```
ffmpeg -i input.mp4 -i output.mp4 -filter_complex “ssim” -report -f null -
```

- `Input.mp4` – source
- `output.mp4` – encoded
- `-filter_complex “ssim”` - calling this filter complex
- `-report` – to record scores; otherwise only appears in Command window
- `-f null -` – tells FFmpeg to output a null file (need –f null -)

SSIM_total.bat

Results in Command Window

```
[Parsed_ssim_0 @ 000001fed2daf0c0] SSIM Y:0.988512 (19.397577) U:0.987286 (18.957099) V:0.988176 (19.272
231) All:0.988252 (19.300226)
```

- FFmpeg displays multiple outputs (Y, U, V, All)
- Correct value is Y (0.98512)
- Second number (19.397577) is SSIM expressed in decibel form which is very seldom used. Here’s the formula
  - -10 * log10 (1 – SSIM)

Compute SSIM After Encoding – Report File

```
[Parsed_ssim_0 @ 0000026f3c3e5f40] SSIM Y:0.988512 (19.397577) U:0.987286 (18.957099) V:0.988176 (19.272231)
All:0.988252 (19.300226)
```

- FFmpeg will create a report named ffmpeg_date_time.log
- Scroll down to the bottom to see the same outputs as the Command window
- Most use Y (0.988512)

Scaling and Computing SSIM

```
ffmpeg -i output_720p.mp4 -i input.mp4 -filter_complex
[0v]scale=1920x1080:flags=lanczos[output_720p];[output _720p][1v]ssim -report -f
null -
```

- `[0v]scale=1920x1080:flags=lanczos[input_720p];` -scale first video `([0v])` to 1080p using lanczos method and label it `input_720p`
- `[input_720p][1v]psnr –` forward it to SSIM using the label `input_720p` and compare it to the first video `[1v]`
- `-report -f null –` as before

SSIM_scale_total.bat

Compute SSIM After Encoding – Frame Scores

```
ffmpeg -y -i output.mp4  -i input.mp4  -lavfi  ssim=output_3M_ssim.log -report -f null -
```

- `input_1080p.mp4` – encoded (note reverse order from previous)
- `input.mp4` – source
- `-lavfi  ssim=output_3MB_ssim.log` - calls Librafilter, computes ssim and inserts individual frame scores into this log file
  - Substitute desired name for name shown
- `-report` – records overall scores; log file only records individual frame scores
- `-f null -` – tells FFmpeg to output a null file (need `-f null -`)

SSIM_Framescores.bat

Compute SSIM After Encoding – SSIM Log

- PSNR log contains individual frame scores
- Can input into Excel/Sheets for additional presentation or analysis

```
n:1 Y:0.990566 U:0.984536 V:0.987105 All:0.988984 (19.579897)
n:2 Y:0.987101 U:0.984367 V:0.987164 All:0.986656 (18.747085)
n:3 Y:0.987141 U:0.984260 V:0.987267 All:0.986682 (18.755629)
n:4 Y:0.985032 U:0.979600 V:0.983576 All:0.983884 (17.927485)
n:5 Y:0.986321 U:0.980962 V:0.984508 All:0.985126 (18.275710)
n:6 Y:0.986133 U:0.981714 V:0.985221 All:0.985245 (18.310517)
n:7 Y:0.984059 U:0.978888 V:0.982913 All:0.983006 (17.697059)
n:8 Y:0.984730 U:0.980108 V:0.984006 All:0.983839 (17.915359)
n:9 Y:0.985227 U:0.979457 V:0.983561 All:0.983987 (17.955363)
n:10 Y:0.983239 U:0.977456 V:0.981956 All:0.982062 (17.462140)
n:11 Y:0.985058 U:0.979351 V:0.983547 All:0.983855 (17.919654)
n:12 Y:0.984303 U:0.978975 V:0.982958 All:0.983191 (17.744554)
n:13 Y:0.983268 U:0.974561 V:0.979834 All:0.981245 (17.268756)
n:14 Y:0.982493 U:0.977151 V:0.982185 All:0.981551 (17.340312)
n:15 Y:0.980892 U:0.974971 V:0.980710 All:0.979875 (16.962647)
n:16 Y:0.978173 U:0.973634 V:0.979811 All:0.977689 (16.514850)
n:17 Y:0.976506 U:0.971259 V:0.978223 All:0.975918 (16.182998)
n:18 Y:0.975514 U:0.972531 V:0.979536 All:0.975687 (16.141651)
n:19 Y:0.975066 U:0.970462 V:0.977965 All:0.974782 (15.982892)
```

Compute SSIM After Encoding – Report File

```
[Parsed_ssim_0 @ 000001318da7ef00] SSIM Y:0.988512 (19.397577) U:0.987286 (18.957099) V:0.988176
(19.272231) All:0.988252 (19.300226)
```

- Same as before

Values for Y

- Which is right?
  - Average score – 0.988512
  - Average and per-frame – 0.988512
  - MSU VQMT - 0.9893891811
- Close enough that it probably doesn't matter
  - Use the same tool/technique to compute for comparison purposes

Questions

- Should be: 2:50

Computing VMAF with FFmpeg – Completely New Section

- Download code here - http://learnffmpeg.s3.amazonaws.com/ffmpeg-vmaf-static-bin.zip
  - This 32-bit version was compiled on November 8, 2019 by Abi Bhat
    - Unzip and run; must use FFmpeg.exe and point to models in this folder
    - **Get all this plus batch and test files if you download zip file**
  - Here are instructions for compiling your own version
    - http://learnffmpeg.s3.amazonaws.com/VMAFintegrationintoFFMPEGframework-081119.pdf
- File requirements
- While encoding – not available
- Post-encode – average score/per-frame score

File Requirements

- Compute during encoding – Not available
- Post-encode
  - Resolution ***must*** be the same (scale before computing)

```
[Parsed_ssim_0 @ 000002799c28e140] Width and height of input videos must be same.
[Parsed_ssim_0 @ 000002799c28e140] Failed to configure input pad on Parsed_ssim_0
Error reinitializing filters!
Failed to inject frame into filter network: Invalid argument
Error while processing the decoded data for stream #1:0
Conversion failed!
```

  - Frame rate ***should*** be the same
    - Unable to produce reliable results with 30 fps source

Scaling and Computing VMAF

- Couldn't make this work

Compute VMAF After Encoding

```
ffmpeg.exe -i output.mp4 -i input.mp4
-lavfi libvmaf="model_path=vmaf_v0.6.1.pkl:log_path=VMAF.txt" -report -f null -
```

- `output.mp4` – encoded (note reverse order from previous)
- `input.mp4` – source
- `-lavfi` - calls Librafilter
- `model_path=vmaf_v0.6.1.pkl – default (4K is vmaf_4k_v0.6.1.pkl)`
- `log_path=VMAF.txt` – log format is XML by default, report is `VMAF.txt`
- `-report` – records overall scores; log file only records individual frame scores
- `-f null -` – tells FFmpeg to output a null file (need `-f null -`)

Other Options

- model_path - Set the model path which is to be used for SVM. Default value: "vmaf_v0.6.1.pkl"
- log_path - Set the file path to be used to store logs.
- log_fmt - Set the format of the log file (xml or json).
- phone_model - Invokes the phone model
- psnr - Enables computing psnr along with vmaf.
- ssim - Enables computing ssim along with vmaf.
- ms_ssim -Enables computing ms_ssim along with vmaf.
- Pool - Set the pool method (mean, min or harmonic mean) to be used for computing vmaf.
- n_threads - Set number of threads to be used when computing vmaf.
- n_subsample -Set interval for frame subsampling used when computing vmaf.
- enable_conf_interval - Enables confidence interval.

Compute VMAF After Encoding – All Metrics

```
ffmpeg.exe -i output.mp4 -i input.mp4 -lavfi libvmaf="model_path=vmaf_v0.6.1.pkl:phone_model=1:lo
ssim=1:log_path=allmetrics.txt" -report -f null -
```

- `output.mp4` – encoded file
- `input.mp4` – source
- `-lavfi` - calls Librafilter
- `model_path=vmaf_v0.6.1.pkl – default` (4K is vmaf_4k_v0.6.1.pkl)
- `psnr=1:ssim=1:ms_ssim=1` – compute psnr, ssim, and ms_ssim
- `Phone_model=1` – Use the phone model (must be default model)
- `log_fmt=xml:log_path=VMAF.txt –` log format is XML, report name is `VMAF.txt`
- `-report` – records overall scores; log file only records individual frame scores
- `-f null -` – tells FFmpeg to output a null file (need `-f null -`)

Log File

metrics.txt - Notepad

File Edit Format View Help


```
<?xml version="1.0"?>
<VMAF version="1.3.15">
        <params model="" scaledWidth="1920" scaledHeight="1080" subsample="1"
num_bootstrap_models="0" bootstrap_model_list_str="" />
        <fyi numOfFrames="120" aggregateVMAF="100" aggregatePSNR="46.2622"
aggregateSSIM="0.998274" aggregateMS_SSIM="0.997082" execFps="3.78692" timeTaken="31.6881" />
        <frames>
                <frame frameNum="0" adm2="0.998586" motion2="0" ms_ssim="0.99935" psnr="51.1215"
ssim="0.999781" vif_scale0="0.939412" vif_scale1="0.997837" vif_scale2="0.999006"
vif_scale3="0.999431" vmaf="100" />
                <frame frameNum="1" adm2="0.992763" motion2="2.02159" ms_ssim="0.997828"
psnr="46.2652" ssim="0.998969" vif_scale0="0.855508" vif_scale1="0.985681" vif_scale2="0.993186"
vif_scale3="0.996201" vmaf="100" />
```


- VMAF phone model score
- PSNR, SSIM, MS SSIM scores

- About 4 fps on HP Zbook notebook; total time 31.7 seconds
- VMAF only about 12.5 seconds

Compute VMAF After Encoding – VMAF Log

- VMAF log contains individual frame scores
- Can input into Excel/Sheets for additional presentation or analysis

output_3MB_psnr.log - Notepad

File Edit Format View Help

```
n:110 mse_avg:1.44 mse_y:1.79 mse_u:0.78 mse_v:0.69 psnr_avg:46.55 psnr_y:45.60
psnr_u:49.21 psnr_v:49.72
n:111 mse_avg:1.51 mse_y:1.88 mse_u:0.82 mse_v:0.72 psnr_avg:46.34 psnr_y:45.39
psnr_u:48.99 psnr_v:49.56
n:112 mse_avg:1.40 mse_y:1.69 mse_u:0.87 mse_v:0.76 psnr_avg:46.67 psnr_y:45.85
psnr_u:48.72 psnr_v:49.33
n:113 mse_avg:1.58 mse_y:1.95 mse_u:0.88 mse_v:0.76 psnr_avg:46.16 psnr_y:45.22
psnr_u:48.70 psnr_v:49.33
n:114 mse_avg:1.52 mse_y:1.88 mse_u:0.84 mse_v:0.73 psnr_avg:46.32 psnr_y:45.39
psnr_u:48.90 psnr_v:49.48
n:115 mse_avg:1.59 mse_y:1.93 mse_u:0.96 mse_v:0.85 psnr_avg:46.12 psnr_y:45.27
psnr_u:48.31 psnr_v:48.86
n:116 mse_avg:1.44 mse_y:1.67 mse_u:1.04 mse_v:0.89 psnr_avg:46.56 psnr_y:45.90
psnr_u:47.94 psnr_v:48.63
n:117 mse_avg:1.65 mse_y:2.02 mse_u:1.01 mse_v:0.85 psnr_avg:45.95 psnr_y:45.08
psnr_u:48.09 psnr_v:48.85
n:118 mse_avg:1.43 mse_y:1.73 mse_u:0.93 mse_v:0.76 psnr_avg:46.57 psnr_y:45.76
psnr_u:48.45 psnr_v:49.33
n:119 mse_avg:1.57 mse_y:1.89 mse_u:1.01 mse_v:0.82 psnr_avg:46.18 psnr_y:45.36
psnr_u:48.08 psnr_v:48.99
n:120 mse_avg:1.42 mse_y:1.56 mse_u:1.23 mse_v:1.01 psnr_avg:46.62 psnr_y:46.20
psnr_u:47.22 psnr_v:48.08
```

Questions

- Should be: 3:00 - Break

VMAF Master

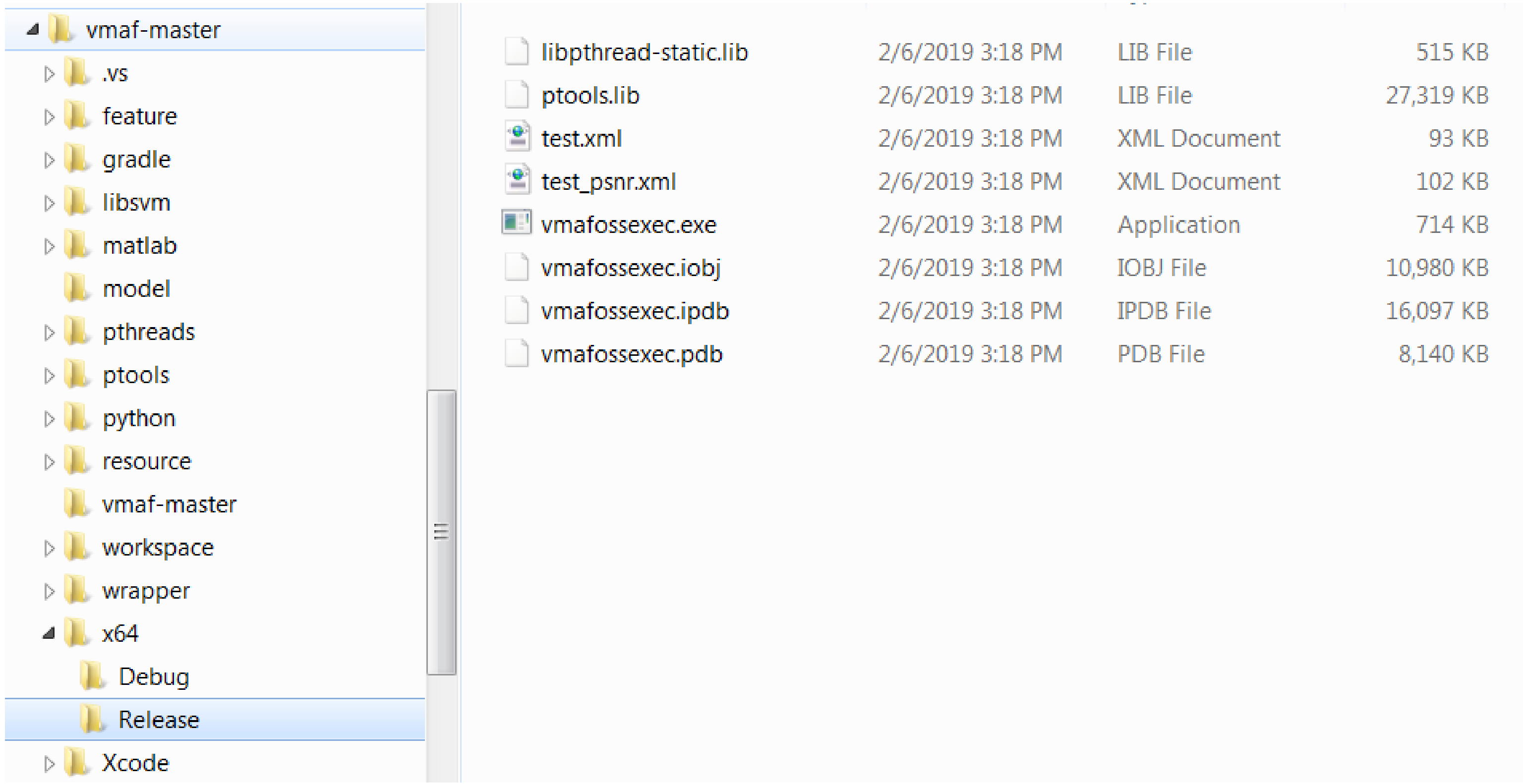
- About
- Installation
- Operation
- Command syntax
- Running the test

About

- VMAF Master
  - Create by Netflix
  - On Github (https://github.com/Netflix/vmaf)
    - Linux only
    - Needs to be compiled
  - Download Windows version at (http://bit.ly/vmafmas)
    - Not the latest version of code
    - Should suffice for most uses
    - Otherwise will have to compile your own

Windows Installation

- Download Windows version at (http://bit.ly/vmafmas)
  - Unzip
  - Copy into c:\vmaf-master folder
    - If not c:\ adjust command lines as needed



Operation

- Need to pre-convert source and distressed (encoded) files to YUV

Command Syntax (page 1)

```
Usage: vmafossexec.exe fmt width height ref_path dis_path model_path
[--log log_path] [--log-fmt log_fmt] [--disable-clip] [--disable-avx]
[--psnr] [--ssim] [--ms-ssim] [--phone-model]
```

- fmt - this identifies the input format of the two video files; must be yuv420p, yuv422p, yuv444p, yuv420p10le, yuv422p10le, yuv444p10le
- width height – you got these
- ref_path - path to the reference file and reference file
- dis_path - path to the “distorted” or compressed video file and file name.
- model_path - path to the model and model. This must be either the default model (vmaf_v0.6.1.pkl) or the 4K model (vmaf_4k_v0.6.1.pkl).
  - You add the phone model via the --phone-model switch shown below. Technically, the phone model is a custom version of the default model, so choose the default model to use the phone model.

Command Syntax (page 2)

```
Usage: vmafossexec.exe fmt width height ref_path dis_path model_path
[--log log_path] [--log-fmt log_fmt][--psnr] [--ssim] [--ms-ssim] [--
phone-model]
```

--log log_path - log file name and path. If no path specified, the log file is stored in the folder with the distorted file
--log-fmt - format for log file (must be either XML or JSON). If you don’t specify, the program stores a CSV file in XML format
--psnr - run the PSNR metric
--ssim - run the SSIM metric
--ms-ssim - run the SSIM metric
--phone-model - run the phone model

Command String

```
C:\vmaf-master\x64\Release\vmafossexec.exe yuv420p 1920 1080
input.yuv output.yuv C:\vmaf-master\model\vmaf_v0.6.1.pkl --psnr --
ssim --ms-ssim --log input_1080p.csv
```

- Program
- Format
- Rez
- Source
- Distressed
- Model
- Metrics
- Log (no path so stored in same folder)
- No log format so stored in CSV format

Complete Script

Scale files to YUV

Delete YUV files

Run metric

scale_vmaf_delete.bat - Notepad

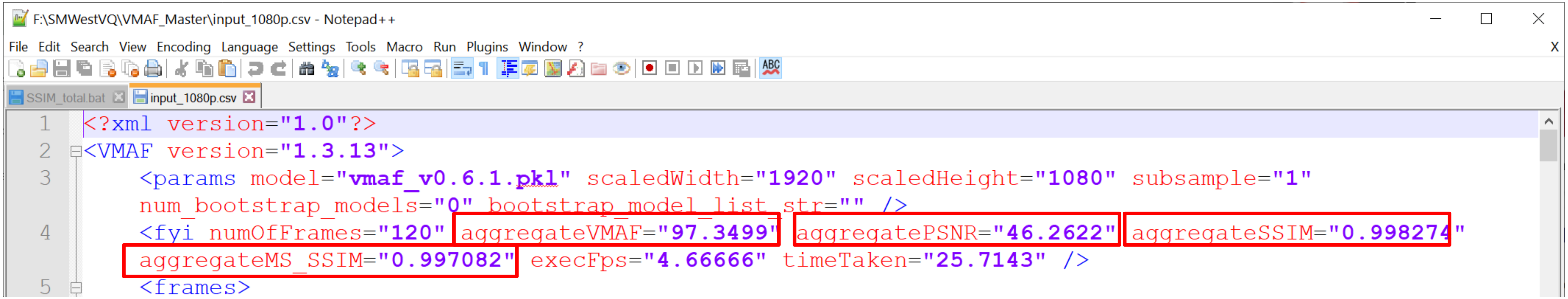
File Edit Format View Help

```
ffmpeg -i input.mp4 -pix_fmt yuv420p -vsync 0  input.yuv
ffmpeg -i output.mp4 -pix_fmt yuv420p -vsync 0  output.yuv

C:\vmaf-master\x64\Release\vmafossexec.exe yuv420p 1920 1080 input.yuv
output.yuv C:\vmaf-master\model\vmaf_v0.6.1.pkl --psnr --ssim --ms-ssim --log
output.csv

del input.yuv
del output.yuv
```

Log File




```
<?xml version="1.0"?>
<VMAF version="1.3.13">
    <params model="vmaf_v0.6.1.pkl" scaledWidth="1920" scaledHeight="1080" subsample="1"
    num_bootstrap_models="0" bootstrap_model_list_str="" />
    <fyi numOfFrames="120" aggregateVMAF="97.3499" aggregatePSNR="46.2622" aggregateSSIM="0.998274"
    aggregateMS_SSIM="0.997082" execFps="4.66666" timeTaken="25.7143" />
    <frames>
```

Questions

- Should be: 3:25

Moscow State University VQMT Tutorials

- Overview
- Loading files
- Choosing metrics
- Working in the Result Plot view
- Multiple instances
- Fixing out of sync videos
- Command line

MSU VQMT Overview

- Where to buy
- Load files
- Choose metric
- Other settings
  - Mask
  - Output
  - Geometry transform
  - Visualization
  - Bad frames
  - Conversion matrix
  - Command line
- Output
  - Plot
    - Options
    - Operation
  - Tabs
  - CSV file
  - JSON

Loading Files into VQMT

- Compatible files
- Incompatible files
  - Convert to Y4M
- Low resolution compatible test files
  - Scaling options
  - Recommendations
- YUV files

Choosing and Configuring Metrics

- VMAF
  - 4K
  - Phone model
  - Both
- PSNR
- SSIM

Strategies for Running Simultaneous Computes

- Add new files and recompute
- Open multiple instances

Scale Videos

```
VQMT  -orig GTAV_30_even.Y4M -in GTAV_V2_Norm_1200.h264 -metr psnr YYUV -csv -resize lanczos to orig
```

`-resize ffmpeg lanczos to orig` - resize using lanczos to size of original, prefer ffmpeg algorithm

`-resize intel lanczos to orig` - resize using lanczos to size of original, prefer intel algorithm

`-resize lanczos to orig` - equivalent to first one

Out of Sync Videos

```
VQMT -orig GTAV_30_even.Y4M 4- -in GTAV_V2_Norm_1200.h264 2- -metr psnr YYUV -csv -
resize lanczos to orig
```

Questions

- Should be: 3:35

SSIMPLUS VOD Monitor Inspector Tool

- Demos:
  - Templates
  - Test
  - Results
  - Comparison

Questions

- Should be: 3:45

Using Metrics: It's Not Just a Number

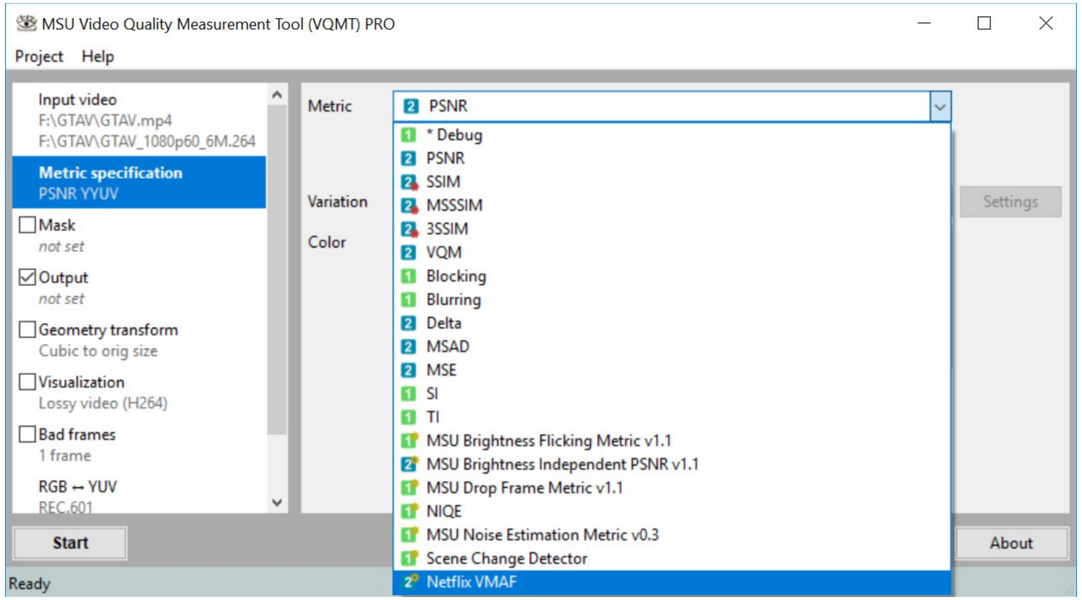
- Comparing codecs/techniques (Rate Distortion Curves/BD-Rate Next)
  - Decisions like:
    - Comparing different encoders
    - Choosing the optimal preset
    - Choosing the optimal bitrate control technique
    - Choosing encoding settings
- Goal
  - To make the best possible decision, not to produce a "number"
  - Single score – interesting, but can be misleading and incomplete
- My analysis technique – leveraging toolset
  - Moscow State University – Video Quality Measurement Tool (VQMT)

Using Objective Benchmarks

- Start with the Number
  - Checking the difference between CBR and Constrained VBR (both 1080p@2500)
    - 200% constrained VBR - 79.28
    - 1-pass CBR - 79.07
  - Both very good, 1-pass CBR cuts encoding time in half, let's use that!
  - OK, let's take a closer look

Moscow State University Video Quality Measurement Tool

- $995
- Free version – 720p < / no command line
- Covered in detail in future lessons
- My tool of choice for low volume comparisons and visual analysis
  - You'll see why in a moment



Then, Look at Results Plot

Let's Look at Frames - Original

Let’s Look at Frames – Constrained VBR

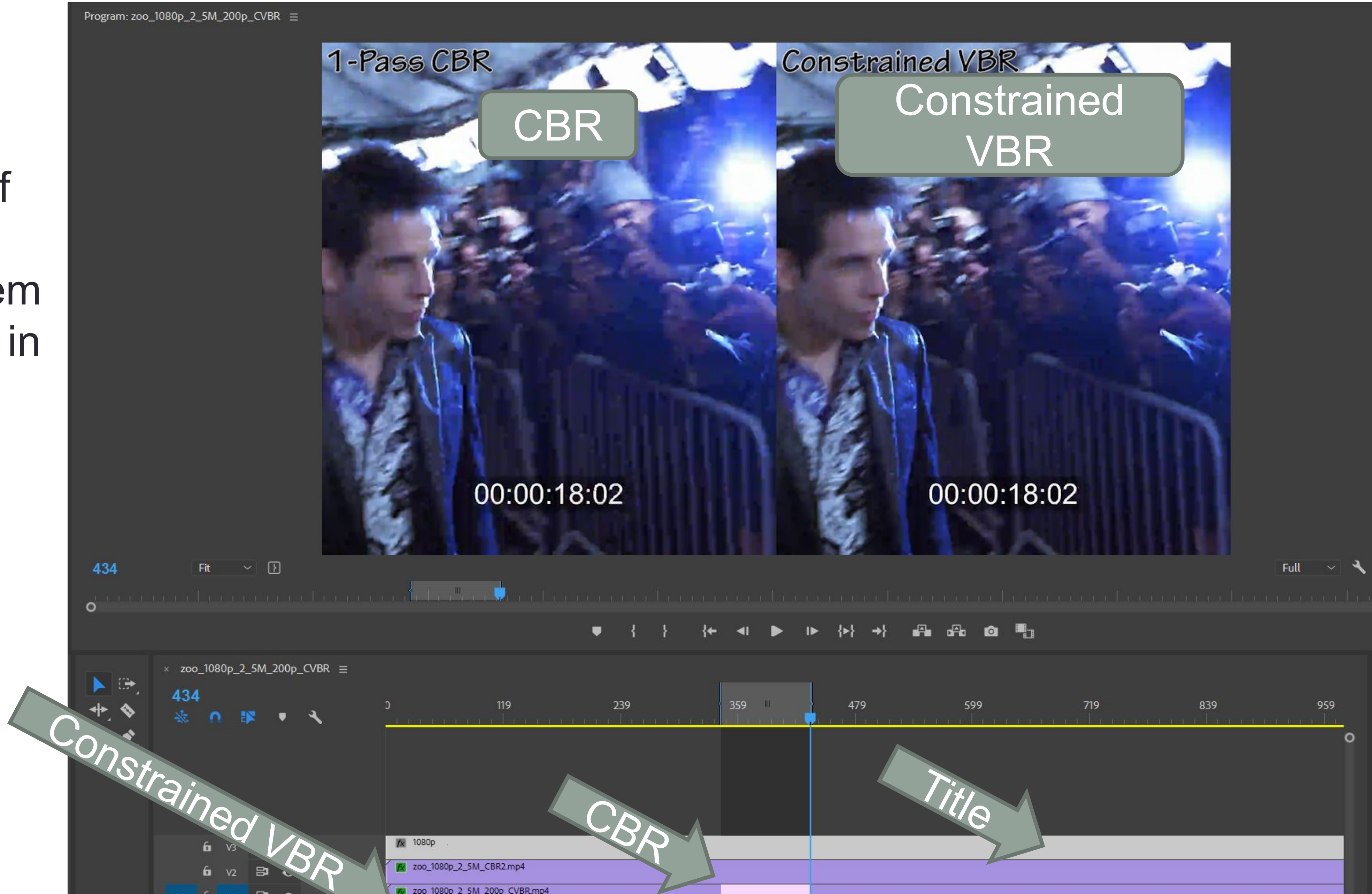
Let's Look at Frames - CBR

But Can You See the Problem In Real Time?

Spikes are very short. Would a viewer even notice?

Load Files into Video Editor

- Load videos to play in and out of timeline
- Verify that problem areas are visible in real time



My Workflow for Encoding Decisions

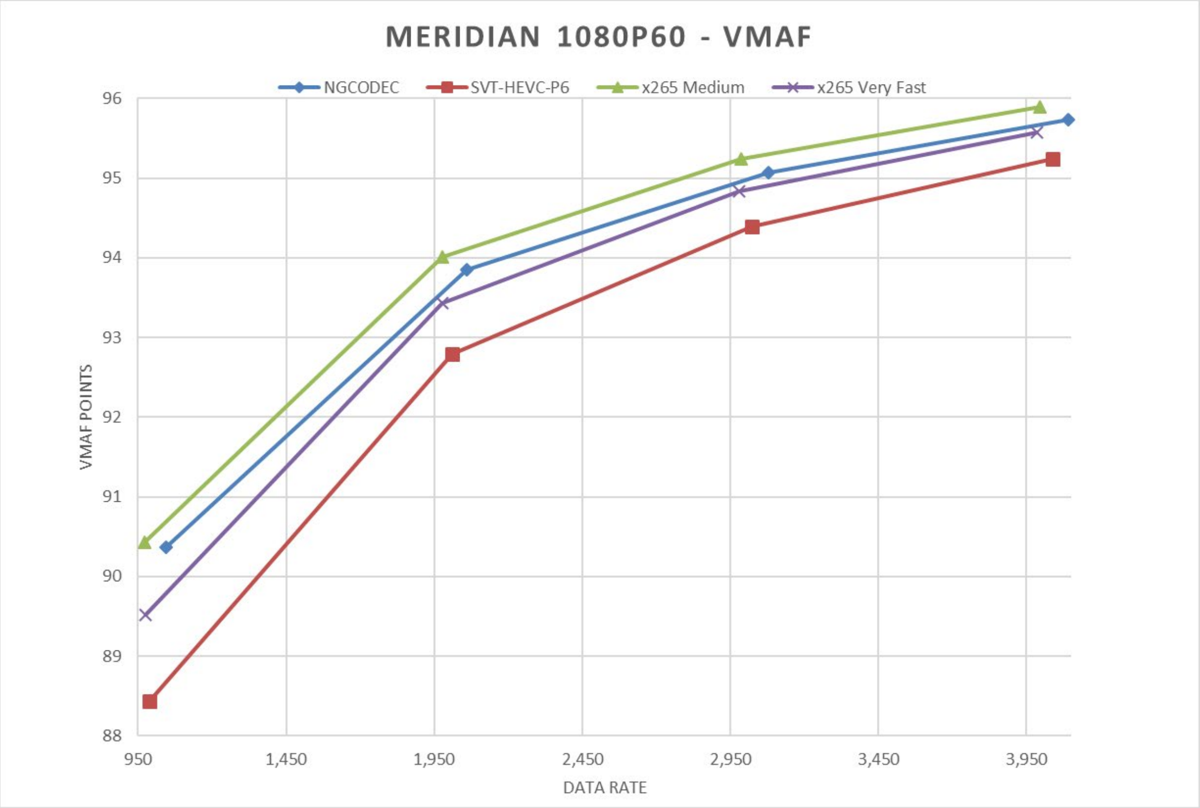
- Run tests
- Review plot
- View bad frames
- Play video to make final determination
- In essence, use metric to identify regions to examine further
  - Never make comparison on the basis of numbers only
  - Always look at frames ***and*** live video

Questions

- Should be: 3:55

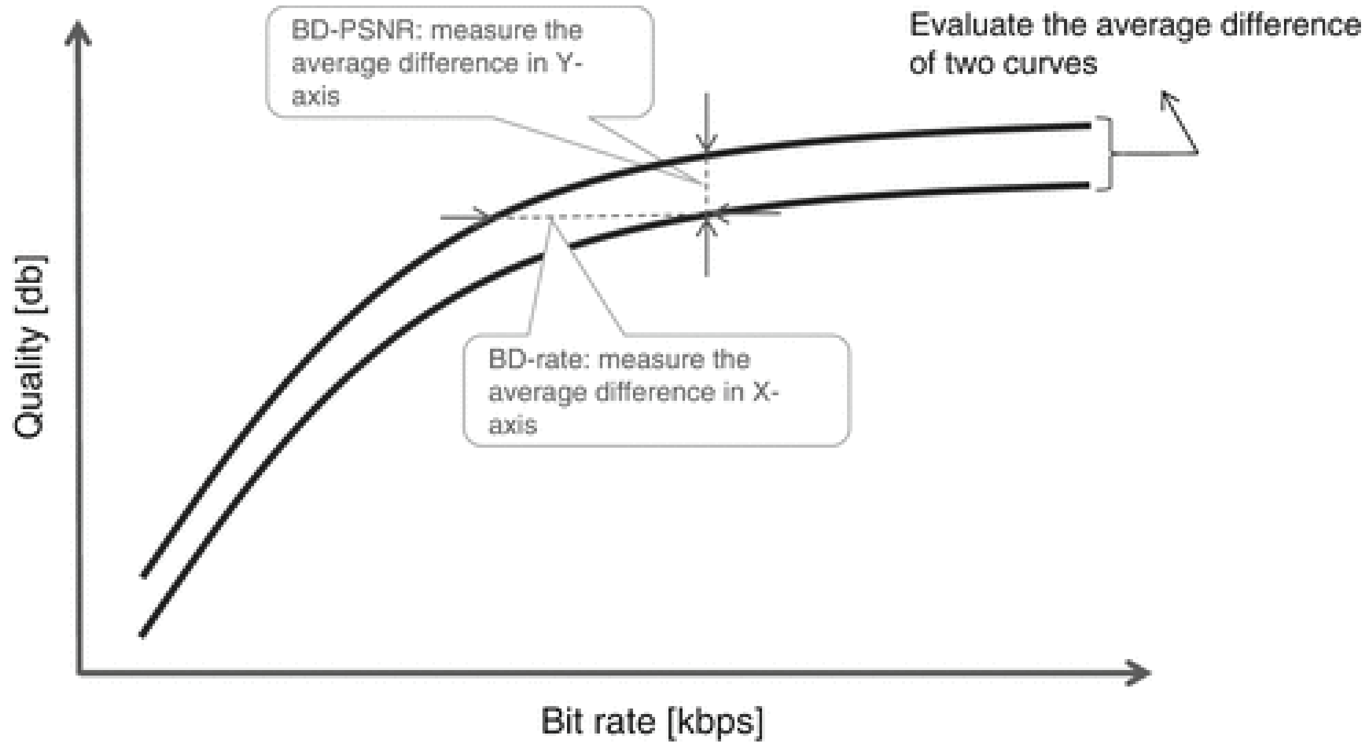
Lesson: Rate Distortion Curves and BD-Rate Functions

- More formal, numbers-only analysis, typically deployed for codec comparisons
- Step 1: Produce "rate-distortion curve"
  - Four encodes with different technologies (VMAF)
    - On right – HEVC transcoders for live broadcasts
  - Rate-distortion curve – how each technology "distorts" at the various data rates



Then Compute Bjontegaard Functions

- Quantifies differences between two curves
  - BD-Rate – data rate saving for the **same quality**
  - BD-PSRN – quality disparity for same **bitrate**
    - Can use with any metric

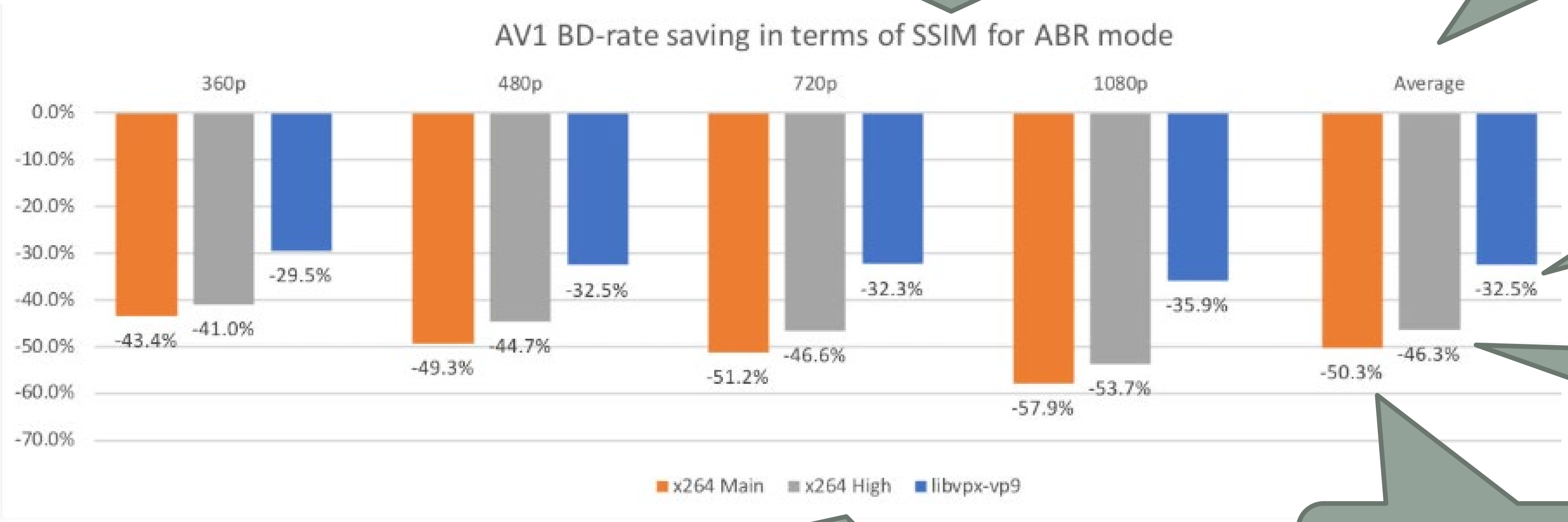


http://bit.ly/BDRPSNR

Facebook AV1 comparisons

Using SSIM

On average, compared to these codecs/settings



32.5% lower data rate than VP9

46.3% lower data rate than x264 High

50.3% lower data rate than x264 Main

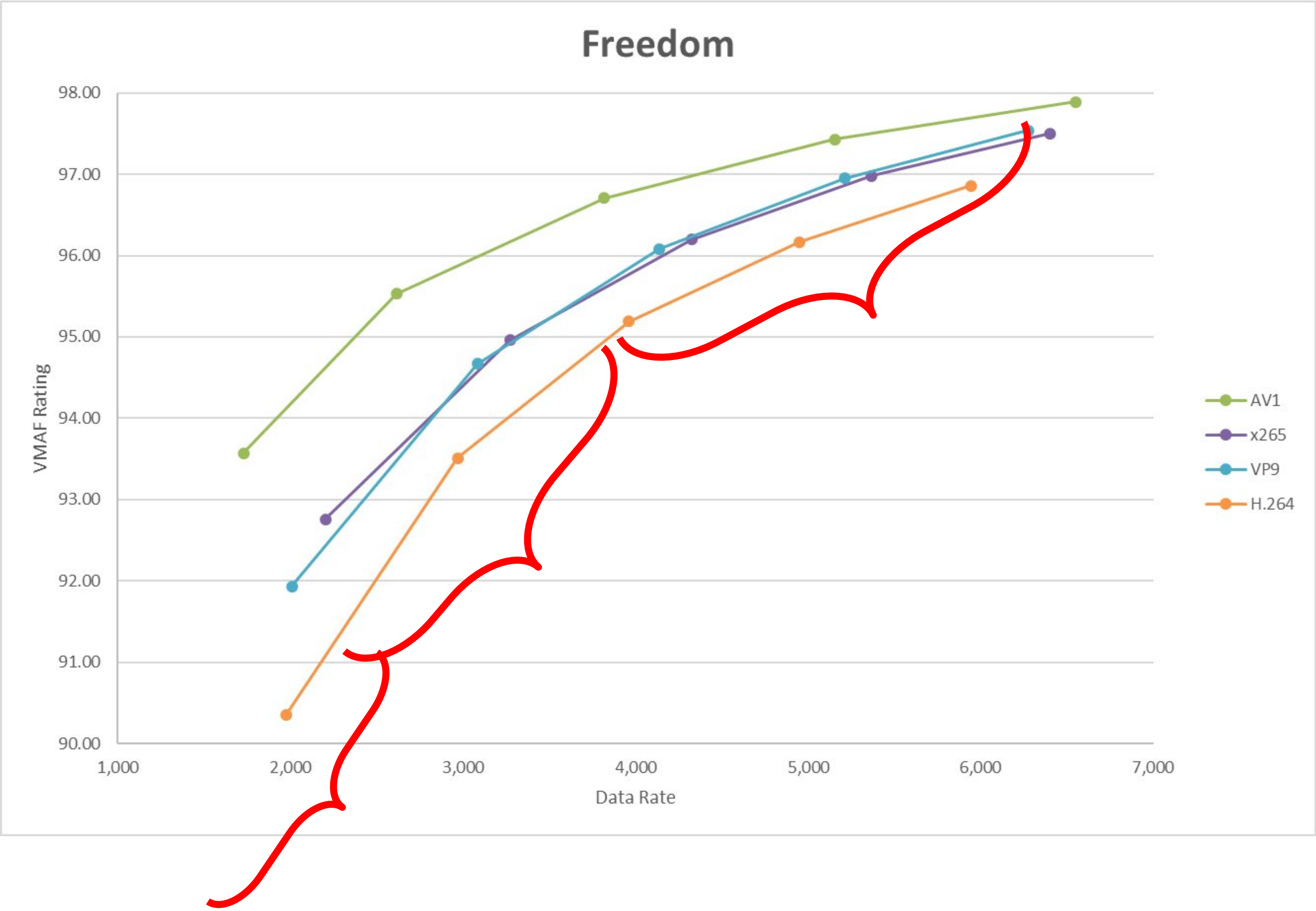
Compared to these codecs/settings

Encoding for Rate Distortion/BD-Rate Analysis

- Need at least four files
- Encoding in realistic quality ranges

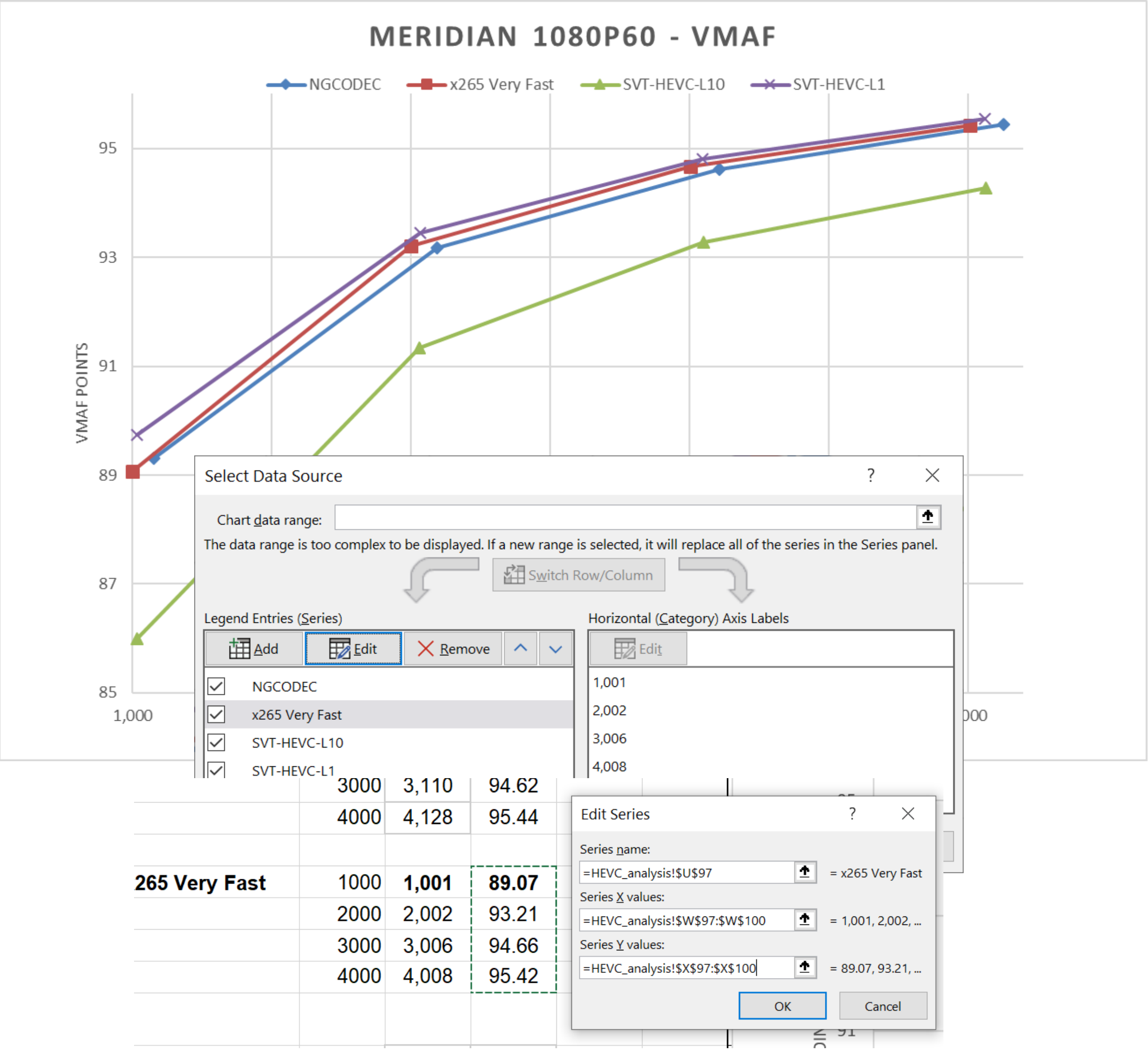
Encoding for Rate Distortion/BD-Rate

- For most relevant results, choose data rates that produce typical quality levels
  - No one cares about this range (95 – 97 VMAF)
  - May be relevant but too small (91 – 94 VMAF)
  - Missing 85-90 which may be relevant
  - Perhaps encode at 1.5, 2, 2.5 and 3 mbps?



Visualization – Rate Distortion Curves

- Overview
  - XLSM file in folder so can reuse
  - Can do in Sheets but Excel clearer and simpler
- Format data
- Create chart
  - Must be scatter with straight lines and markers
- Insert data
- Customize graph area
- Rinse and repeat

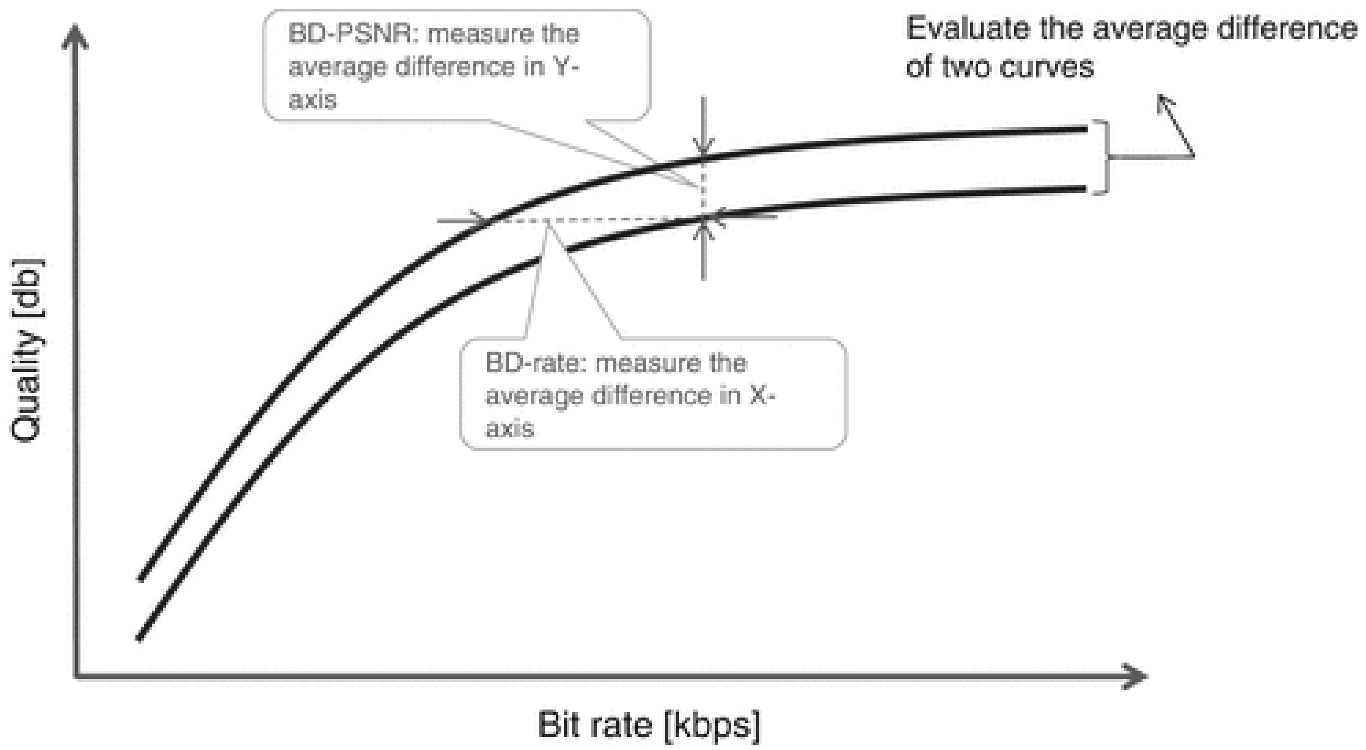


BD-Rate Functions

- For more information
  - bit.ly/BD_functions
- Review – what BD-rate functions are
- Using macro from Tim Bruylants, ETRO, Vrije Universiteit Brussel
- Excel spreadsheet with macro is available for download in the lesson folder

Review

- BD-Rate - Average data rate saving for same quality
  - Cited much more often
- BD-PSNR- Average quality differential at same data rate



http://bit.ly/BDRPSNR

Macro 1 – BD-RATE

- Always referential and have to pick the reference
  - Here, SVT is reference
  - Result – On average, NGCodec can produce same quality as SVT at data rate reduction of 4.21%
- BDBR macro
  - Blue – bitrate of reference file (SVT)
  - Red – metric score of reference file (SVT)
  - Purple – bitrate of target file (NGCodec)
  - Green – metric score of target file (NGCodec)

NGCODEC				SVT-HEVC-P6		
Dinnerscene - 1080p	Bitrate	VMAF	BDRate	Dinnersc ene -	Bitrate	VMAF
1000	1,029	80.20	-4.21	1000	970.1	79.22
2000	2,029	87.68		2000	1971	87.00
3000	3,029	90.75	BD Quality	3000	2970	90.17
4000	4,029	92.48	0.39	4000	3970	91.94

NGCODEC				SVT-HEVC-P6		
Dinnerscene - 1080p	Bitrate	VMAF	BDRate	Dinnersc ene -	Bitrate	VMAF
1000	=BDBR(AF3:AF6,AG3:AG6,$AB3:$AB6,$AC3:$AC6)					79.22
2000	2,029	87.68		2000	1971	87.00
3000	3,029	90.75	BD Quality	3000	2970	90.17
4000	4,029	92.48	0.39	4000	3970	91.94

Round Robin Presentation

VMAF	NGCODEC	SVT-HEVC-P6	x265 Medium	x265 Very Fast
NGCODEC	X	-4.21	12.28	-5.19
SVT-HEVC-P6	4.40	X	17.45	-1.07
x265 Medium	-10.93	-14.86	X	-15.73
x265 Very Fast	5.48	1.08	18.66	X

Macro 2 – BD-PSNR (BD-Quality)

- Always referential and have to pick the reference
  - Here, SVT is reference
  - Result – at all data rates, NGCodec's quality averages .39 VMAF points better than SVT
- BDSNR macro
  - Blue – bitrate of reference
  - Red – metric score of reference
  - Purple – bitrate of comparison
  - Green – metric score of comparison

NGCODEC				SVT-HEVC-P6		
Dinnerscene - 1080p	Bitrate	VMAF	BDRate	Dinnersc ene -	Bitrate	VMAF
1000	1,029	80.20	-4.21	1000	970.1	79.22
2000	2,029	87.68		2000	1971	87.00
3000	3,029	90.75	BD Quality	3000	2970	90.17
4000	4,029	92.48	0.39	4000	3970	91.94

NGCODEC				SVT-HEVC-P6		
Dinnerscene - 1080p	Bitrate	VMAF	BDRate	Dinnersc ene -	Bitrate	VMAF
1000	1,029	80.20	-4.21	1000	970.1	79.22
2000	2,029	87.68		2000	1971	87.00
3000	3,029	90.75	BD Quality	3000	2970	90.17
=BDSNR(AF3:AF6,AG3:AG6,$AB3:$AB6,$AC3:$AC6)						

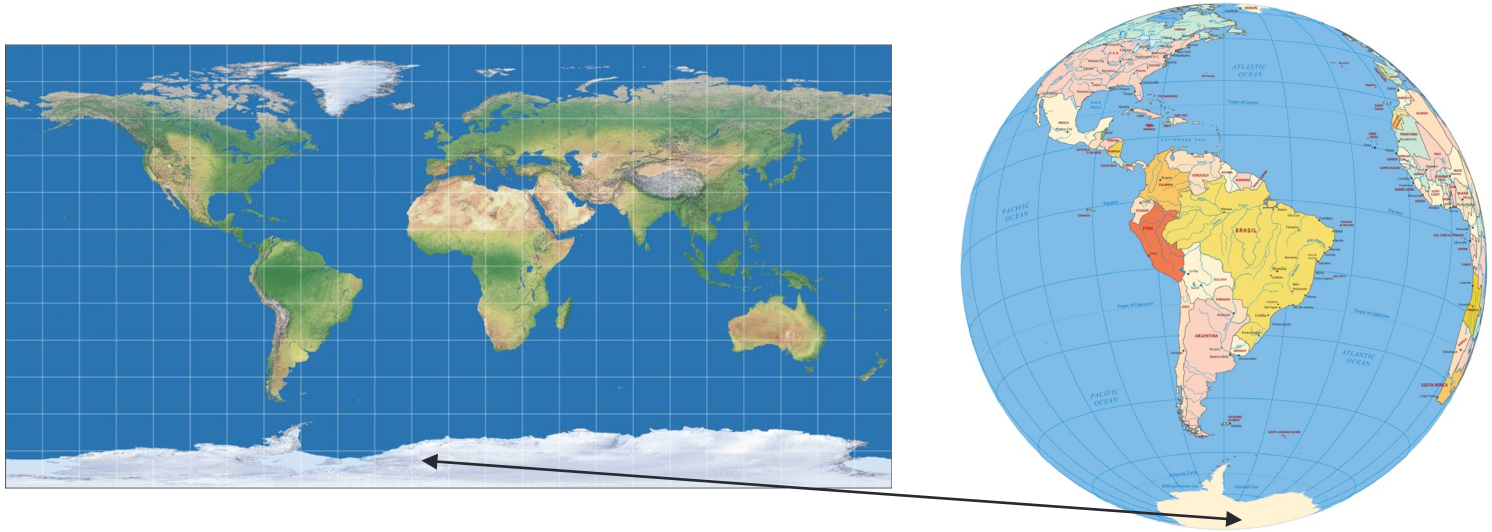
Questions

- Should be: 4:10

What about VR?

- The problem
- Solutions
- The workaround

The Problem



- Multiple VR storage formats
  - Equirectangular above is most common
    - Heavily distorted at poles
  - All represent 360 image in flat world
- VR is 360
  - Relatively similar in the middle
  - Heavily distorted at poles

Issues

- General
  - Where is viewer looking?
    - Is this relevant?
    - Can we weight by presumed focus of attention?
    - Should we?

- General
  - Do flat metrics work?
    - If so, which?
  - What VR metrics are available?
    - Do they work?

Tools and Metrics

- There are multiple VR metrics
  - They are not generally accessible
  - None in MSU, SQM, or Hybrik

Reviews are Mixed

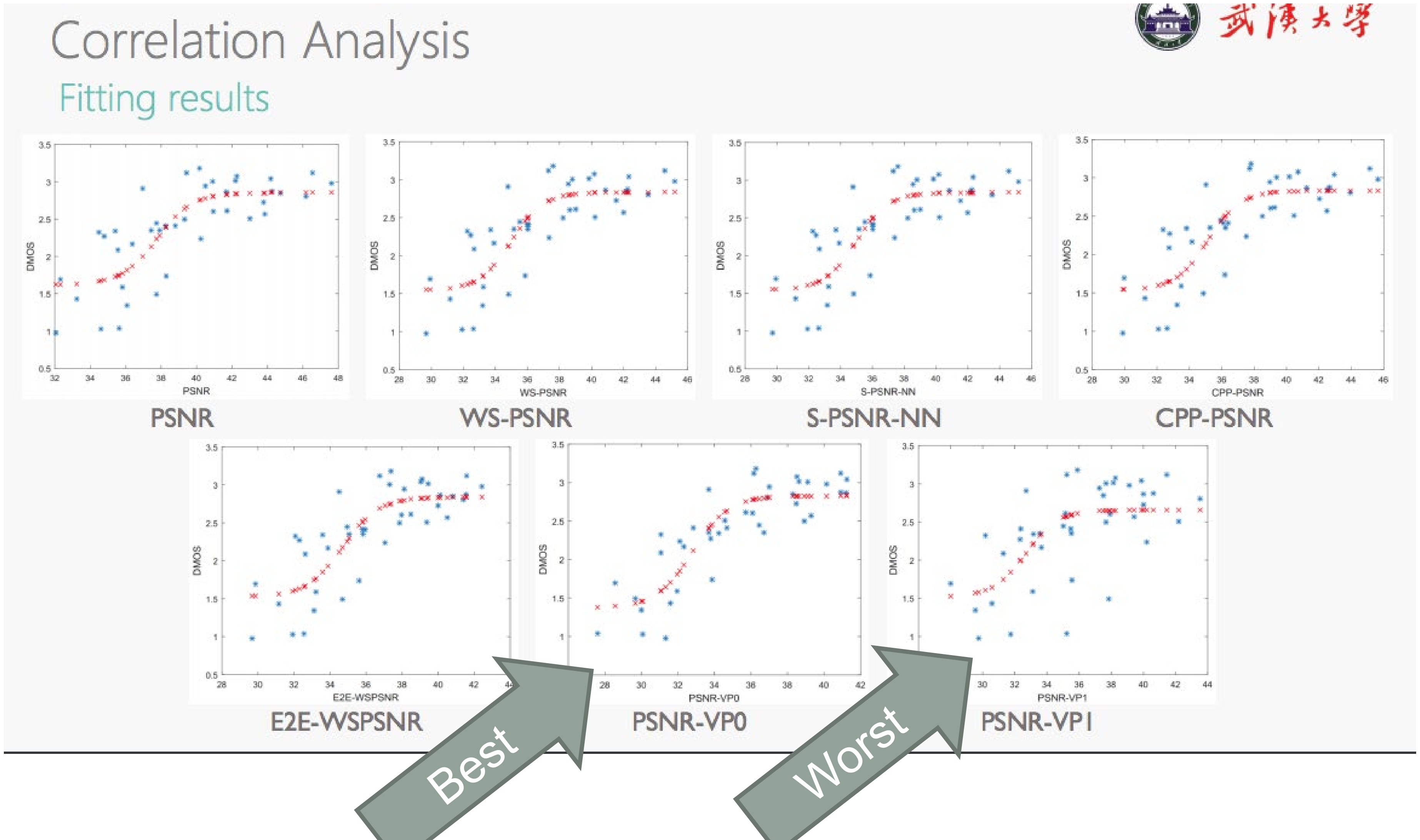
- On the Performance of Objective Metrics for Omnidirectional Visual Content (http://bit.ly/vrqm_1), "Objective metrics specifically designed for 360-degree content ***do not outperform conventional methods*** designed for 2D images."
- An evaluation of quality metrics for 360 videos (http://bit.ly/vrqm_2), "Most objective quality measures are well correlated with subjective quality. Among the evaluated quality measures***, [traditional flat] PSNR is the most appropriate for 360 video communications."***
- Weighted-to-Spherically-Uniform Quality Evaluation for Omnidirectional Video (http://bit.ly/vrqm_3), "Our method makes the quality evaluation results ***more accurate and reliable since*** it avoids error propagation caused by the conversion from resampling representation space to observation space."

Benchmarking Virtual Reality Video Quality Assessement (http://bit.ly/vrqm_4)

Objective model	Description
PSNR	Peak Signal-to-Noise Ratio. Calculates PSNR based on all samples with equal weight.
WS-PSNR	Weighted to Spherically uniform PSNR. Calculates PSNR based on all samples with a weighted parameter, depending on the sample area on the spherical surface.
S-PSNR-NN	Spherical PSNR without interpolation. Calculates PSNR based on a point set evenly sampled on the sphere surface, whose value is taken from the nearest neighbor integer sample positions to avoid the impact due to interpolation filters.
CPP-PSNR	PSNR for Carster Parabolic Projection. Compares quality across different projection methods using Carster Parabolic Projection format.
E2E-WSPSNR	End to End WS-PSNR. Proposes end to end assessment for comparing compression performance of different projection.
PSNR-VP0 and PSNR-VP1	Calculates PSNR on 2D displays with the two viewports (VPs) rendered from the decoded bit stream with predefined parameters.

- Evaluated these metrics

Benchmarking Virtual Reality Video Quality Assessement (http://bit.ly/vrqm_4)

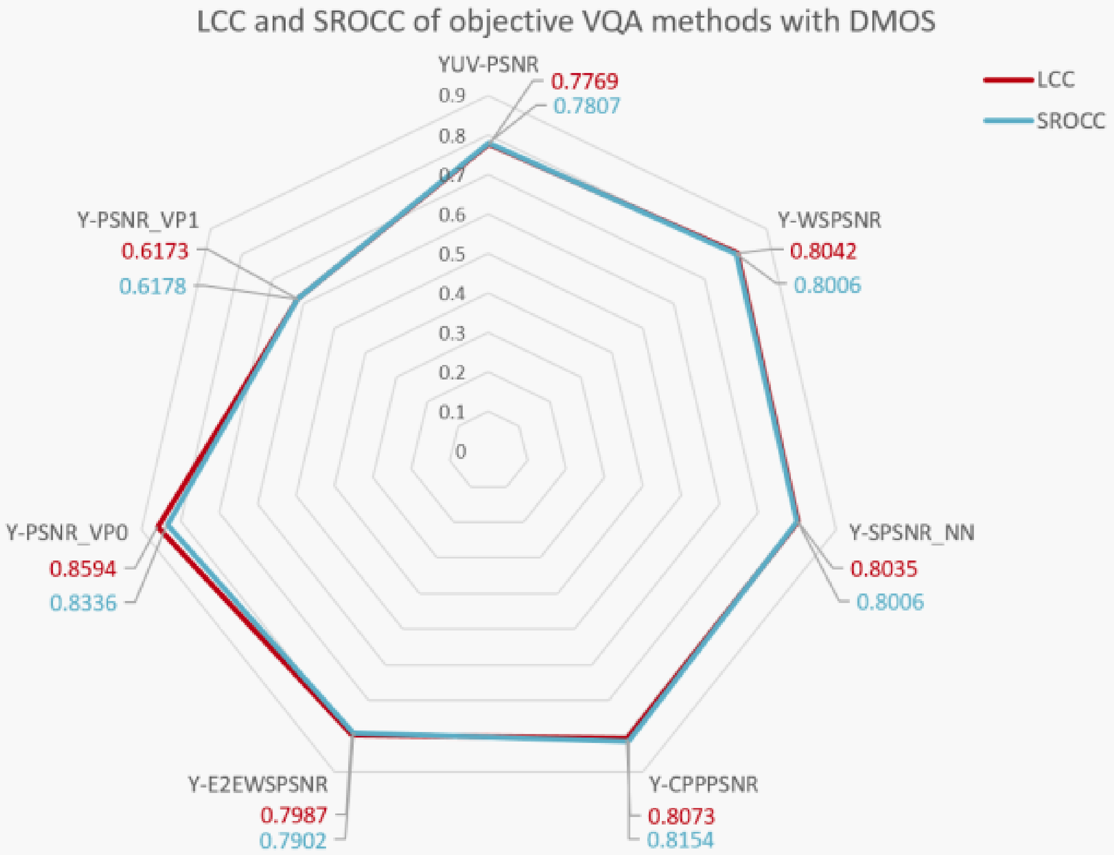


Benchmarking Virtual Reality Video Quality Assessement (http://bit.ly/vrqm_4)

Not night and day difference

Higher is Better

Algorithm	LCC	SROCC
PSNR	0.7769	0.7807
WS-PSNR	0.8042	0.8006
S-PSNR-NN	0.8035	0.8006
CPP-PSNR	0.8073	0.8154
E2E-WSPSNR	0.7987	0.7902
PSNR-VP0	**0.8594**	**0.8336**
PSNR-VP1	0.6173	0.6178



What I've Done

- All work performed for Pixvana; ***data courtesy Pixvana***
  - Compared Samsung WS-PSNR with PSNR and VMAF
- Focus
  - Utility for choosing appropriate data rate for switching resolutions in ABR ladder
    - Less convenient than PSNR/VMAF
    - Is it worth the effort

- Compared Samsung WS-PSNR with PSNR and VMAF
  - https://github.com/Samsung/360tools

Supported formats

```
+       Equirectangular projection (ERP)
+       Icosahedral projection (ISP)
+       Octahedron projection (OHP)
+       Cubemap projection (CMP)
+       Truncated Square Pyramid projection (TSP)
+       Segmented Sphere Projection (SSP)
+       Reshaped Icosahedral projection (RISP)
+       Reshaped Octahedron projection (ROHP)
+       Reshaped Cubemap projection (RCMP)
```

Supported quality metrics

```
+       PSNR - conventional Peak Signal to Noise Ratio quality metrics
+       S-PSNR - spherical PSNR (requires sphere_655362.txt file with point coordinates)
+       WS-PSNR - weighted Spherical PSNR (for equirectangular projection only)
+       CPP-PSNR - equal area common projection PSNR
```

Building Encoding Ladder

- Netflix-like method
  - Top rate determined by budget or minimum quality
  - Lower data rates distributed by formula (so rungs between 1.5/2x apart)
  - Use quality metric to choose resolution at each rate
  - Did WS-PSNR provide substantially different result than PSNR

Zap1 - VMAF	4K	2K	1080p	720p	480p	360p	240p
5000	90.19	89.70	84.82				
4500	89.58	88.23	84.38				
4000	88.43	87.50	83.84				
3800	87.88	87.14	83.58				
3600	87.27	86.71	83.25				
3400	86.60	85.72	82.87				
3200	85.80	85.40	82.45				
3000	85.03	85.09	82.01				
2800	83.97	84.34	81.43				
2600	82.86	83.50	80.85				
2400	81.45	82.51	80.09	71.92			
2200	79.79	81.24	79.20	71.35			
2000	77.94	79.82	78.04	70.66			
1800		78.11	76.73	69.70	53.28		
1600		75.91	74.93	68.41	52.82		
1400		73.26	72.64	66.89	52.13	32.07	
1200		69.83	69.69	64.68	51.05	31.75	
1000		65.15	65.75	61.64	49.36	31.17	
900		62.26	63.25	59.64	48.11	30.76	
800		58.69	60.27	57.20	46.54		
700		54.29	56.62	54.06	44.63	29.18	
600		48.79	52.32	50.65	42.02	27.84	
500			46.65	45.96	38.74	25.92	
400			39.06	40.23	34.21	23.11	
300			28.52	32.68	x		
200			13.88	21.73	x		

Building Encoding Ladder

	Video 1			Video 2			Video 3		
	VMAF	PSNR	WS-PSNR	VMAF	PSNR	WS-PSNR	VMAF	PSNR	WS-PSNR
4K > 2K	8,000	5,000	5,000	3,000	2,200	2,000	8,000	5,000	5,000
2K > 1080p	3,200	2,000	1,800	1,000	900	900	3,200	2,000	1,800
1080p > 720p	1,000	1,000	1,000	400	500	400	1,000	1,000	1,000
720p > 480p	NA	500	500	NA	100	100	NA	500	500
480p > 360p	NA	300	200	NA	NA	NA	NA	300	200

- Not really
- Three different files
  - Switch points very different between VMAF and PSNR/WS-PSNR
- On these three files, however, PSNR/WS-PSNR deliver about the same result
- Conclusion: PSNR/VMAF both more accessible, faster, so WS-PSNR adds no value in this application

Voronoi-Based Testing

2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)

Voronoi-based Objective Quality Metrics for Omnidirectional Video

- Researchers from Trinity College in Dublin Ireland
- Divide video into patches using the spherical Voronoi diagram of M evenly distributed points on the sphere
- Encoded six ODV (omni-directional video) test files encoded at various resolutions and data rates (each 10 seconds long)
- Measured subjective ratings
- Measured objective with multiple techniques both 2D and ODV
- Measured correlation

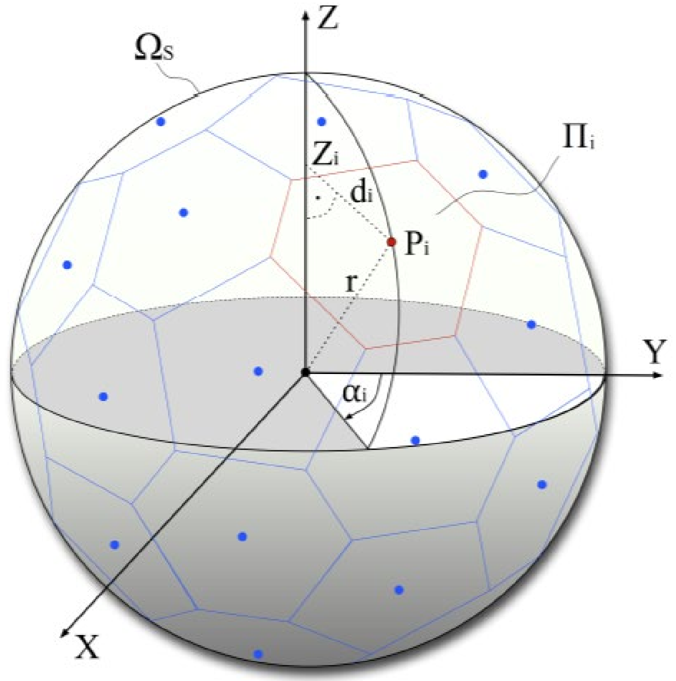


Fig. 1: Spherical Voronoi diagram.

Equirectangular vs. Cubemap Image Formats

- How camera stores image
- Projected to 360-degree display from both sources

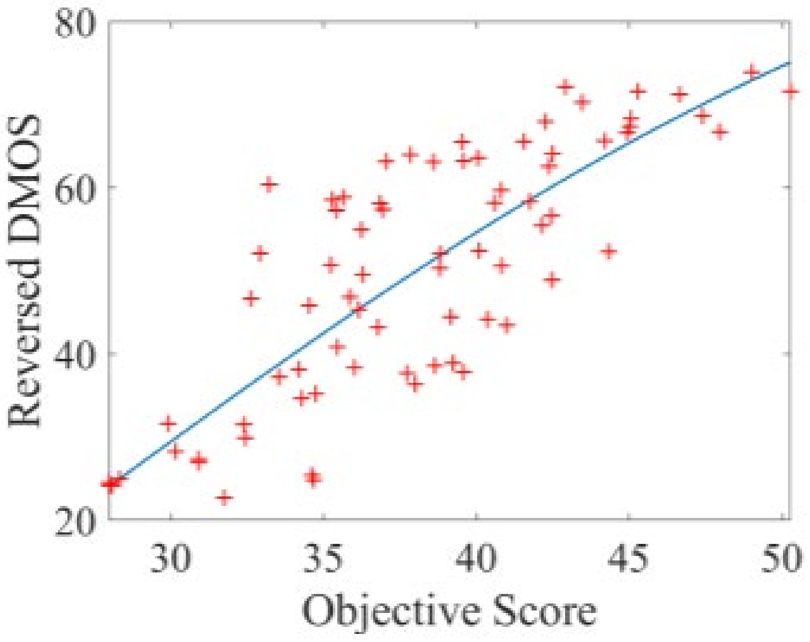
- Equirectangular is more popular and had more support
- Cubemap has less distortion

Correlations

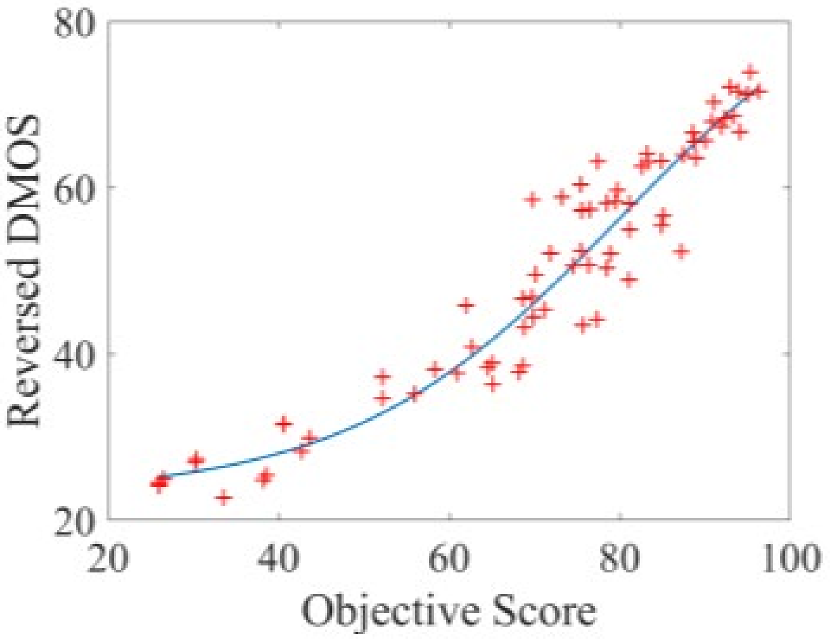
ERP – equirectangular
CMP – cube mapped (more accurate)

	Metrics	Representation	PLCC	SROCC	RMSE	MAE
2D metric	PSNR	ERP	0.8292	0.7979	8.7921	7.0102
	PSNR	CMP	0.8429	0.8101	8.4822	6.7224
ODV metric	S-PSNR-I	ERP	0.8479	0.8139	8.3675	6.5937
	S-PSNR-NN	ERP	0.8489	0.8150	8.3432	6.5718
	WS-PSNR	ERP	0.8485	0.8141	8.3519	6.5790
	CPP-PSNR	ERP	0.8479	0.8136	8.3690	6.5954
2D metric	SSIM	ERP	0.7347	0.7107	10.5253	8.5131
	SSIM	CMP	0.7419	0.7209	10.4370	8.5427
2D metric	MS-SSIM	ERP	0.9085	0.8888	6.6162	5.3242
	MS-SSIM	CMP	0.9125	0.8954	6.4904	5.1064
2D metric	VMAF	ERP	0.9160	0.8861	6.2562	4.7724
	VMAF	CMP	0.9267	0.8998	5.9792	4.4919
ODV metric	VI-PSNR	ERP	0.8545	0.8251	8.1746	6.4750
	VI-SSIM	ERP	0.8132	0.7968	9.1138	7.2579
	VI-MS-SSIM	ERP	0.9447	0.9334	5.2625	4.2398
	VI-VMAF	ERP	**0.9661**	**0.9499**	**4.2356**	**3.1269**

Correlation with Subjective



(a) PSNR in ERP format.



(b) VI-VMAF

Among all the metrics considered in this paper, the one with the best performance is VI-VMAF.

Reality Check

2D-VMAF – 9267

VI-VMAF – 0.9661

Metrics	Representation	PLCC	SROCC	RMSE	MAE
PSNR	ERP	0.8292	0.7979	8.7921	7.0102
PSNR	CMP	0.8429	0.8101	8.4822	6.7224
S-PSNR-I	ERP	0.8479	0.8139	8.3675	6.5937
S-PSNR-NN	ERP	0.8489	0.8150	8.3432	6.5718
WS-PSNR	ERP	0.8485	0.8141	8.3519	6.5790
CPP-PSNR	ERP	0.8479	0.8136	8.3690	6.5954
SSIM	ERP	0.7347	0.7107	10.5253	8.5131
SSIM	CMP	0.7419	0.7209	10.4370	8.5427
MS-SSIM	ERP	0.9085	0.8888	6.6162	5.3242
MS-SSIM	CMP	0.9125	0.8954	6.4904	5.1064
VMAF	ERP	0.9160	0.8861	6.2562	4.7724
VMAF	CMP	0.9267	0.8998	5.9792	4.4919
VI-PSNR	ERP	0.8545	0.8251	8.1746	6.4750
VI-SSIM	ERP	0.8132	0.7968	9.1138	7.2579
VI-MS-SSIM	ERP	0.9447	0.9334	5.2625	4.2398
VI-VMAF	ERP	**0.9661**	**0.9499**	**4.2356**	**3.1269**

Available on Github

- Metric source code
- Python script for running the metric

Voronoi-based VMAF for Omnidirectional Video

Requirements

Current implementation is based python version 2.

First, you need to install the following dependencies:

- pip install wget
- pip install imageio
- pip install python-csv

Second, you need to add all the distorted and reference mp4 files into the **videos** folder.

Test

- python 360vmaf.py --w 3840 --h 2160 --f 100 --r sounders2

> --w: resolution width of the videos --h: resolution height of the videos --f: number of frames --r: reference (original) .mp4 video name

Results will be located in the project folder with distorted video name and in *.csv* format

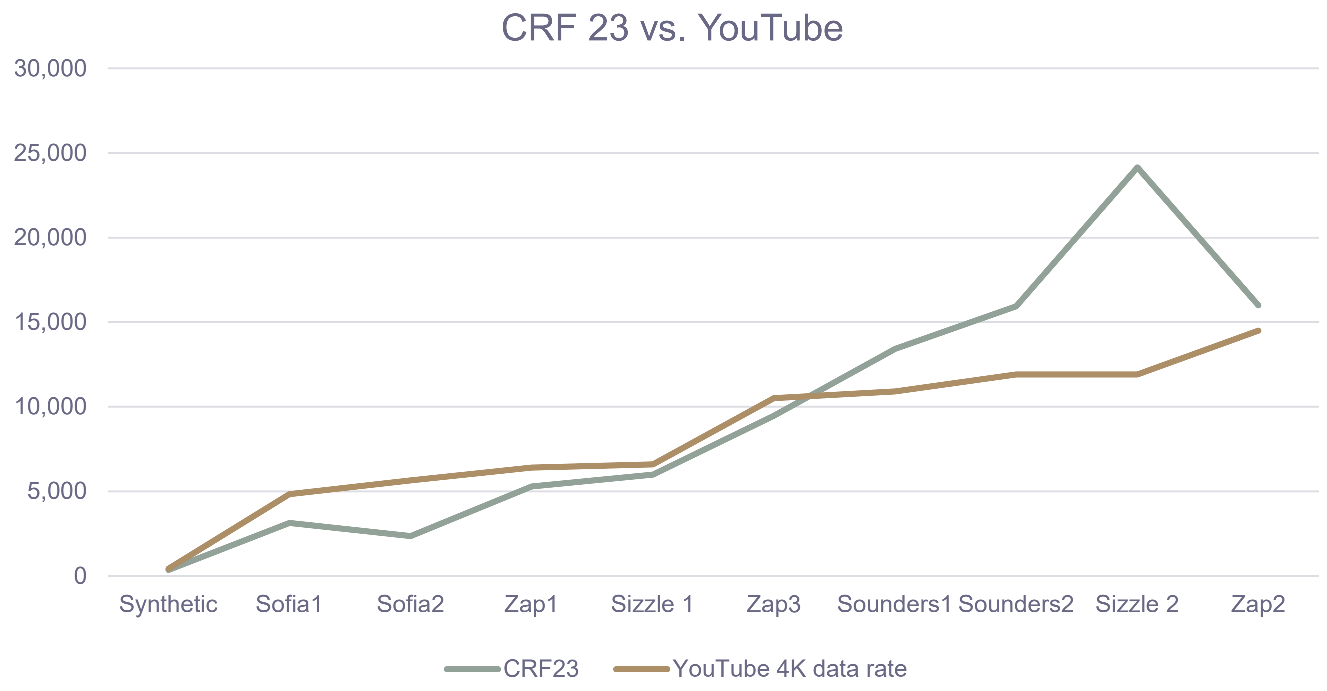
https://github.com/V-Sense/VI_VMAF_4_360

What about VR? VR Videos at CRF 23

CRF 23 - H264	Synthetic	Sofia1	Sofia2	Zap1	Sizzle 1	Zap3	Sounders1	Sounders2	Sizzle 2	Zap2
240p	25	56	60	98	86	112	274	311	125	146
360p	41	111	112	191	170	257	552	587	311	358
480p	59	178	173	309	283	452	902	936	625	683
720p	96	353	332	621	590	1,009	1,796	1,902	1,712	1,623
1080p	153	724	639	1,287	1,252	2,321	3,670	4,058	4,968	3,964
2K	214	1,240	1,022	2,175	2,205	3,981	5,981	6,952	10,344	6,863
4K	**355**	**3,129**	**2,348**	**5,286**	**5,987**	**9,467**	**13,431**	**15,934**	**24,159**	**15,999**

- Equirectangular format
  - Ran CRF 23 across multiple resolutions
  - Videos ranged from very simple animations to highly detailed videos
- 4K data rates ranged from 1.15 to 24.1 Mbps
- Per-title absolutely essential to VR

CRF 23 Compared to YouTube



- Similar pattern
- One very major diversion
- CRF 23 averaged about 1.25 Mbps higher
- Remove outlier and delta averaged 25 kbps

Pixvana Verification of VMAF/PSNR

- Create 5 versions of each full rez VR file to be viewed in order
- Center file is CRF 23 value
- Other files vary in intervals of 3 VMAF points
  - File 1 – 87 VMAF
  - File 2 – 90 VMAF
  - File 3 – 93 VMAF
  - File 4 – 96 VMAF
  - File 5 – 99 VMAF
- Tests ~ 20 viewers
  - Choose lowest quality file that's commercial grade (floor)
  - Choose file at which you see no meaningful improvement (ceiling)

Finding Lowest Acceptable 1080p Quality

Video Name	Average	Standard Deviation	Calculate Data Rate	CRF 23 Data Rate	Delta
Sofia1	1.67	0.71	2,136	3,129	46.49%
Zap1	2.24	1.11	4,056	5,286	30.33%
Sizzle1	2.43	1.05	4,746	5,987	26.15%
Sounders1	2.38	1.33	7,760	13,431	73.08%
Zap3	2.9	0.97	8,750	9,467	8.19%
Average			**5,490**	**7,460**	**35.89%**
Remove outlier			**4,922**	**5,967**	**21.24%**

- CRF 23 averaged 35.89% higher than floor selected by viewers
  - One major outlier
- Was always high, not low
  - Might produce too high a data rate, but in 100% of cases, exceeded floor, so always produced “acceptable” quality

Which Metric? VMAF or PSNR

- VMAF ranged from 90 - 95.5; PSNR from 37.8 - 48.3
- VMAF has much less dispersion and lower standard deviation
- ***Much lower*** standard deviation as percentage of average
- VMAF more accurate than PSNR
- Rule of thumb:
  - CRF 23 s/deliver 93 VMAF or higher
  - If 93 VMAF (again) should be acceptable quality
  - Same for 43.5 PSNR, but less accurate tool

Video Name	Calculate Data Rate	VMAF Calc DR	PSNR Calc DR
Sofia1	2,136	95.5	48.3
Zap1	4,056	93.5	43.6
Sizzle1	4,746	94	45.4
Sounders1	7,760	89.9	37.9
Zap3	8,750	92.0	42.3
Average		**93.0**	**43.5**
Standard Deviation		**2.128**	**3.856**
As percentage of average		**2.29%**	**8.86%**

Once You Have Highest it Becomes Math Exercise

- Step 1: Choose highest
- Step 2: Choose lowest
- Step 4: fill in the blanks (between 150/200% apart)

200 kbps

500 kbps

1000 kbps

1600 kbps

2100 kbps

3100 kbps

4600 kbps

Then Question is:

- Netflix approach
  - Compute VMAF scores at multiple resolutions at each data rate
  - Choose best quality at each resolution
  - VMAF proven for 2D by Netflix, what about 3D?

Zap1 - VMAF	4K	2K	1080p	720p	480p	360p	240p
5000	90.19	89.70	84.82				
4500	89.58	88.23	84.38				
4000	88.43	87.50	83.84				
3800	87.88	87.14	83.58				
3600	87.27	86.71	83.25				
3400	86.60	85.72	82.87				
3200	85.80	85.40	82.45				
3000	85.03	85.09	82.01				
2800	83.97	84.34	81.43				
2600	82.86	83.50	80.85				
2400	81.45	82.51	80.09	71.92			
2200	79.79	81.24	79.20	71.35			
2000	77.94	79.82	78.04	70.66			
1800		78.11	76.73	69.70	53.28		
1600		75.91	74.93	68.41	52.82		
1400		73.26	72.64	66.89	52.13	32.07	
1200		69.83	69.69	64.68	51.05	31.75	
1000		65.15	65.75	61.64	49.36	31.17	
900		62.26	63.25	59.64	48.11	30.76	
800		58.69	60.27	57.20	46.54		
700		54.29	56.62	54.06	44.63	29.18	
600		48.79	52.32	50.65	42.02	27.84	
500			46.65	45.96	38.74	25.92	
400			39.06	40.23	34.21	23.11	
300			28.52	32.68	x		
200			13.88	21.73	x		

What about VR

- Ran tests on three files testing top 3 switch points
  - Test different resolutions at that switch point
- Three comparisons
  - Pick best quality or even
  - Round 1 – low res file should win (VMAF 3 higher)
  - Round 2 – should be even (at switch point)
  - Round 3 – high res file should win (VMAF 3 higher)

Clip	Zap1 (dining room/kitchen)		
Encoding complexity	Moderate (CRF 23 = 5,286)		
	VMAF to Subjective	Average	Error
4K to 2K			
Round 1 (2 should win)	Accurate	1.73	
Round 2 (should be tie)	Accurate	1.5	
Round 3 (1 should win)	Accurate	1	
2K to 1080p			
Round 1 (2 should win)	Accurate	1.58	
Round 2 (should be tie)	Accurate	1.45	
Round 3 (1 should win)	Accurate	1.08	
1080p to 720p			
Round 1 (2 should win)	Accurate	1.9	
Round 2 (should be tie)	OK	1.22	Hi Rez
Round 3 (1 should win)	Accurate	1.08	
Low Round	2K-Accurate	2.29	

Overall

- In 2 of 3 trials, worked beautifully (correct 14 out of 15 trials)
- In third trial, incorrect 5 of nine
- But! Highest resolution file always won
  - More testing may be performed, but
  - If close to switch point, go with higher resolution

Clip	Sounders 1 (Stadium)		
Encoding complexity	Complex (CRF 23 = 13,431)		
	VMAF to Subjective	Average	Error
4K to 2K			
Round 1 (2 should win)	Inaccurate	1.25	Hi Rez
Round 2 (should be tie)	Accurate	1.42	NA
Round 3 (1 should win)	Accurate	1.17	NA
2K to 1080p			
Round 1 (2 should win)	Inaccurate	1.38	Hi Rez
Round 2 (should be tie)	Inaccurate	1.07	Hi Rez
Round 3 (1 should win)	Accurate	1	NA
1080p to 720p			
Round 1 (2 should win)	Inaccurate	1.17	Hi Rez
Round 2 (should be tie)	Inaccurate	1.15	Hi Rez
Round 3 (1 should win)	Accurate	1.14	NA
Low Round	2K - Inaccurate	1.73	

Evolve This Into an Encoding Strategy

- Create different ladders based upon complexity
- Allocate videos based upon CRF 23 score
- Create different ladders for different codecs (H.264/HEVC)

H264 Ladders (SWAG)

Under 5 Mbps

Rung	Rez	Data Rate
1	4K	5,000
2	4K	3,400
3	4K	2,200
4	2K	1,500
5	2K	1,000
6	1080p	700
7	1080p	500
8	720p	300

5 – 10 Mbps

Rung	Rez	Data Rate
1	4K	10,000
2	4K	6,500
3	4K	4,000
4	4K	3,000
5	2K	2,000
6	2K	1,300
7	2K	900
8	1080p	600
9	1080p	400
10	720p	300

10 – 20 Mbps

Rung	Rez	Data Rate
1	4K	20,000
2	4K	13,000
3	4K	8,500
4	4K	5,500
5	4K	3,500
6	2K	2,400
7	2K	1,600
8	1080p	1000
9	1080p	600
10	720p	400

20+ Mbps

Rung	Rez	Data Rate
1	4K	30,000
2	4K	18,000
3	4K	11,000
4	2K	7,000
5	2K	4,500
6	2K	3,000
7	2K	2,000
8	1080p	1,200
9	1080p	800
10	720p	500
11	720p	300

HEVC Ladders (SWAG)

Under 5 Mbps

Rung	Rez	Data Rate
1	4K	4,500
2	4K	3,000
3	4K	2,000
4	4K	1,200
5	4K	800
6	2K	500
7	1080p	300

5 – 10 Mbps

Rung	Rez	Data Rate
1	4K	10,000
2	4K	6,000
3	4K	4,000
4	4K	2,500
5	4K	1,500
6	2K	1,000
7	2K	600
8	1080p	400
9	1080p	300

10 – 20 Mbps

Rung	Rez	Data Rate
1	4K	20,000
2	4K	12,000
3	4K	8,000
4	4K	5,000
5	4K	3,000
6	4K	2,000
7	2K	1,200
8	1080p	800
9	1080p	500
10	720p	300

20+ Mbps

Rung	Rez	Data Rate
1	4K	30,000
2	4K	20,000
3	4K	13,000
4	4K	8,500
5	4K	5,500
6	4K	3,500
7	2K	2,200
8	1080p	1,500
9	1080p	1,000
10	720p	600
11	720p	400

VR – Preliminary Observations

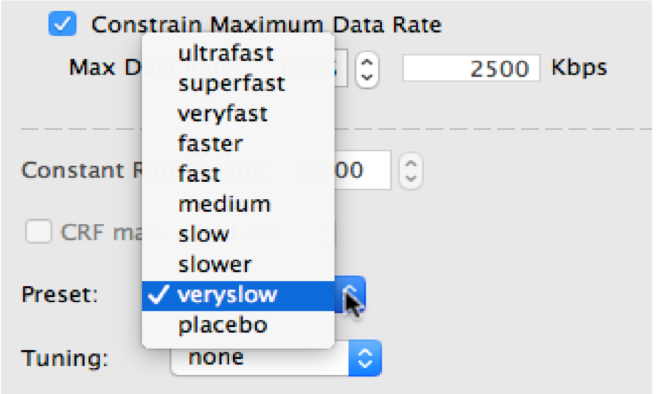
- Different storage formats (equirectangular vs. cubemap vs. diamond plane) will impact quality at a given data rate more than any encoding parameter or technique
  - Equirectangular appears to lag behind cube mapping (as an example)
- Though VMAF/CRF seem reasonably well proven for equirectangular, haven’t confirmed similar effectiveness for other storage formats

Questions

- Should be: 4:30

Choosing the Optimal Encoding Time/Quality Tradeoff

- All encoders/codecs have configuration option that trades off time vs. quality
  - This technique lets you choose the best option
- Here – looking at x264 presets. What are presets?
  - Simple way to adjust multiple parameters to trade off encoding speed vs. quality
  - Used by virtually all x264 encoders
  - Medium is generally the default preset



When to Use This Technique

- When evaluating new encoders
- When choosing/evaluating encoding settings
- When comparing codecs

Test Procedure

- Choose test files
  - 1 movie (Tears of Steel)
  - 2 animations (Sintel, BBB)
  - Two general purpose (concert, advertisement)
  - One talking head
  - Screencam
  - Tutorial (PPT/Video)
- 2. Encode to all presets targeting around 96 VMAF max
  - All files encoded to different bitrates
- 3. Measure encoding time
- 4. Measure Average VMAF
- 5. Measure Low-Frame VMAF

Average VMAF

Average Quality	Ultrafast	Superfast	Veryfast	Faster	Fast	Medium	Slow	Slower	Veryslow	Placebo	Total Delta
Tears of Steel	89.20	92.00	93.29	95.45	95.59	96.22	96.43	96.56	96.67	96.65	8.38%
Sintel	88.29	92.66	93.85	95.84	95.99	96.38	96.56	96.68	96.83	96.75	9.68%
Big Buck Bunny	87.26	91.26	92.68	95.03	95.29	95.53	95.75	95.87	96.05	96.01	10.08%
Talking Head	95.19	92.55	93.66	94.90	94.86	95.18	95.29	95.43	95.51	95.39	3.20%
Freedom	91.95	91.15	92.63	94.58	94.51	95.37	95.59	95.84	96.15	96.04	5.48%
Haunted	91.30	88.61	89.43	91.30	91.08	91.98	92.08	92.35	92.49	92.45	4.38%
Screencam	90.92	92.56	93.52	94.75	94.75	94.70	94.77	94.86	94.92	94.91	4.41%
Tutorial	93.42	94.66	95.55	96.16	96.17	96.17	96.26	96.28	96.29	96.10	3.07%
Average	**90.53**	**91.37**	**92.59**	**94.52**	**94.56**	**95.11**	**95.28**	**95.46**	**95.62**	**95.55**	**6.08%**

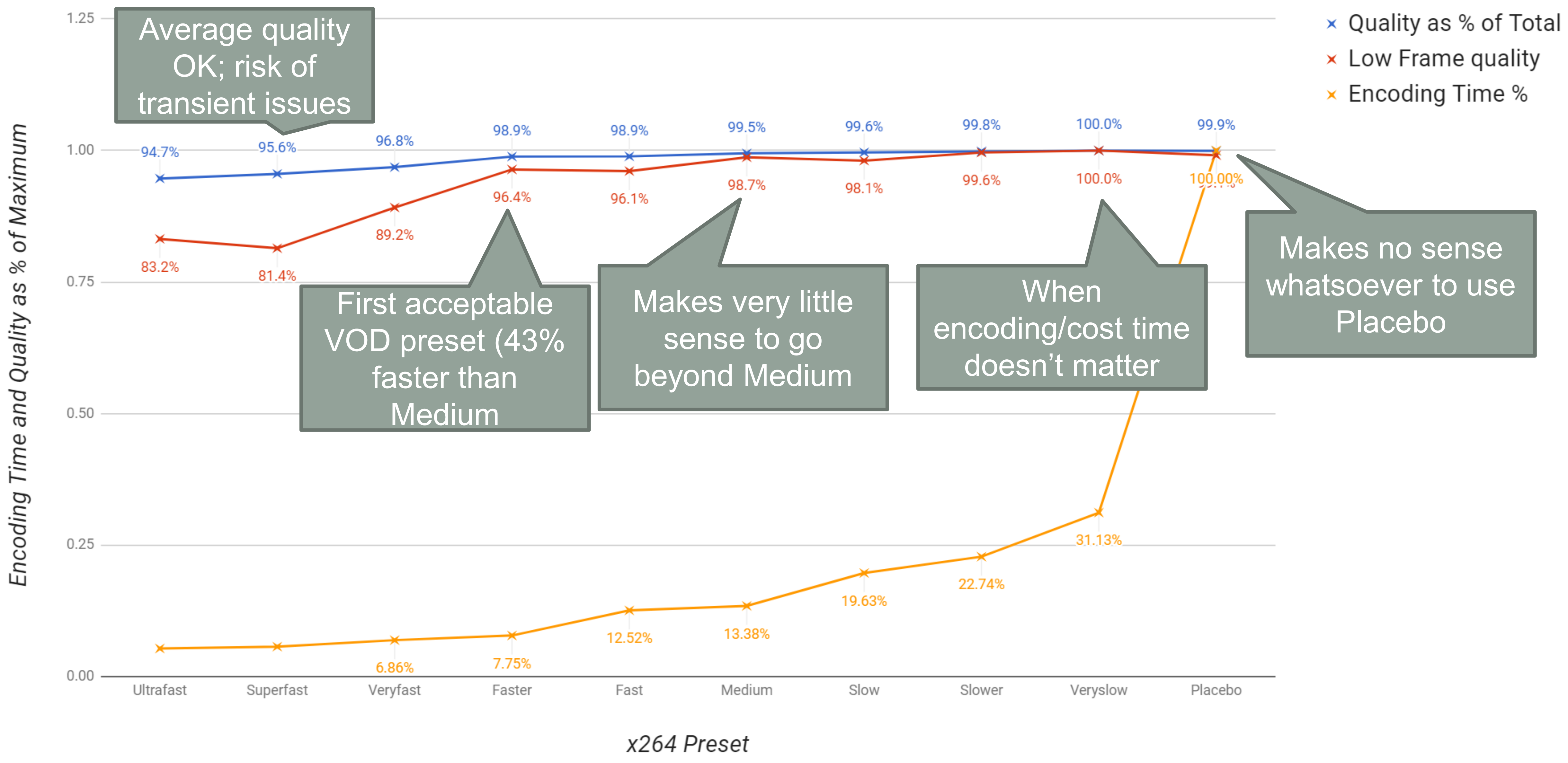
- Red is lowest quality
- Green highest quality
- Note top values – average 95.62 (not Placebo)
- Very slow averages best quality
  - But only 8% spread between best and worst

Low-Frame VMAF

Low Frame Quality	Ultrafast	Superfast	Veryfast	Faster	Fast	Medium	Slow	Slower	Veryslow	Placebo	Total Delta
Tears of Steel	70.16	74.82	77.67	84.51	85.02	85.34	85.44	86.38	85.33	85.10	23.12%
Sintel	68.77	69.79	74.93	79.12	80.41	82.27	81.90	82.98	84.89	82.61	23.45%
Big Buck Bunny	55.42	65.11	62.50	79.33	79.57	82.70	79.18	83.22	80.24	79.08	50.15%
Talking Head	88.90	61.43	88.53	91.62	91.32	92.11	92.03	92.49	92.16	91.37	50.56%
Freedom	76.49	82.79	83.96	87.59	87.29	88.72	89.00	89.35	90.28	90.05	18.03%
Haunted	60.36	57.18	62.69	64.62	61.63	67.33	67.74	68.64	72.08	72.28	26.42%
Screencam	56.16	68.53	71.00	76.39	77.44	77.06	78.04	79.26	78.04	75.21	41.12%
Tutorial	85.68	90.99	91.95	94.11	94.24	94.68	94.50	94.21	94.02	70.58	34.15%
Average	**70.02**	**68.52**	**75.05**	**81.13**	**80.88**	**83.08**	**82.55**	**83.84**	**84.16**	**83.41**	**33.37%**

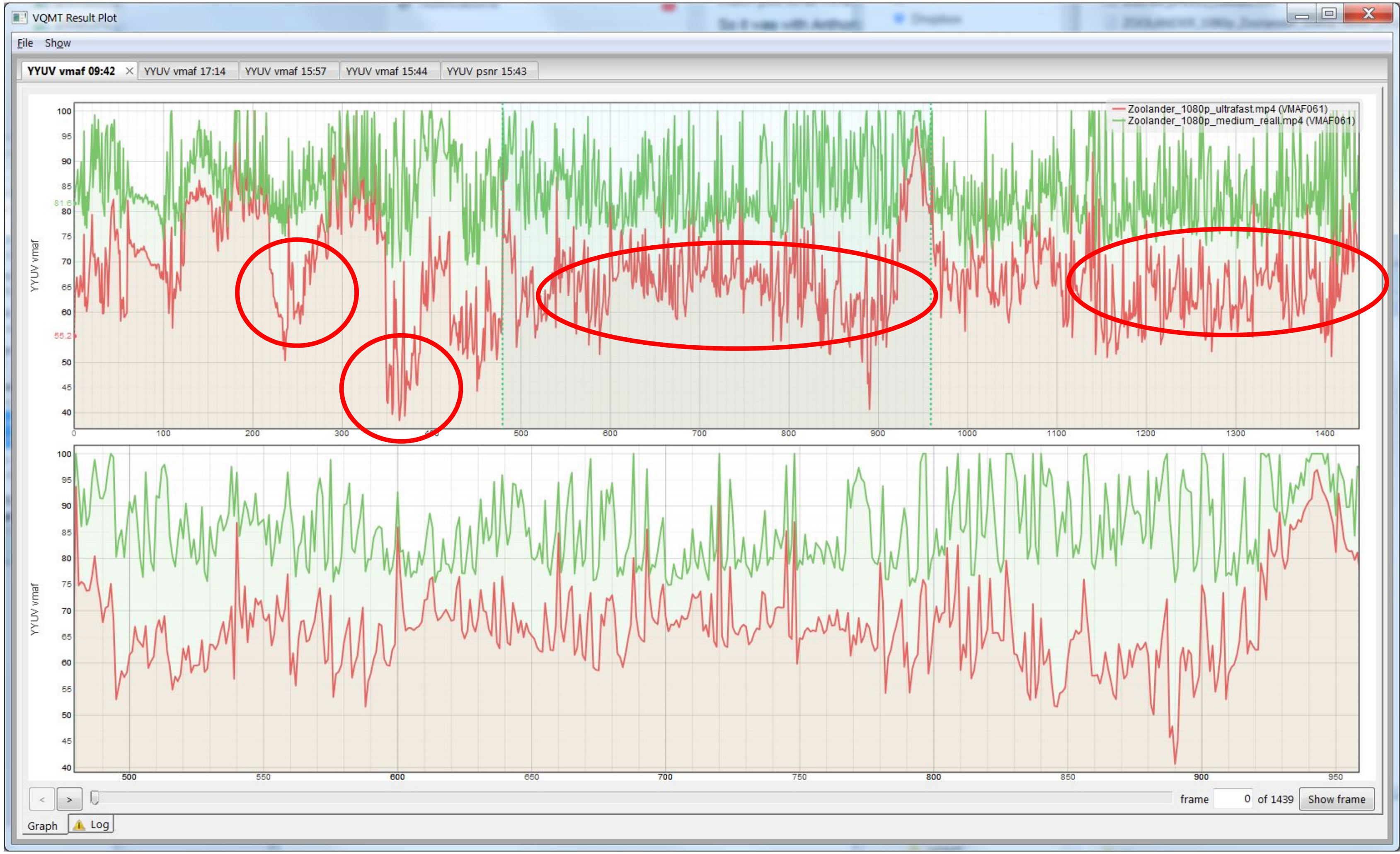
- Red is lowest quality
- Green highest quality
- Note top values – average 84.16 (not Placebo)
- Very slow averages best quality
  - 33% spread between best and worst

Average Quality, Low-Frame Quality and Encoding Time Per x264 Presets



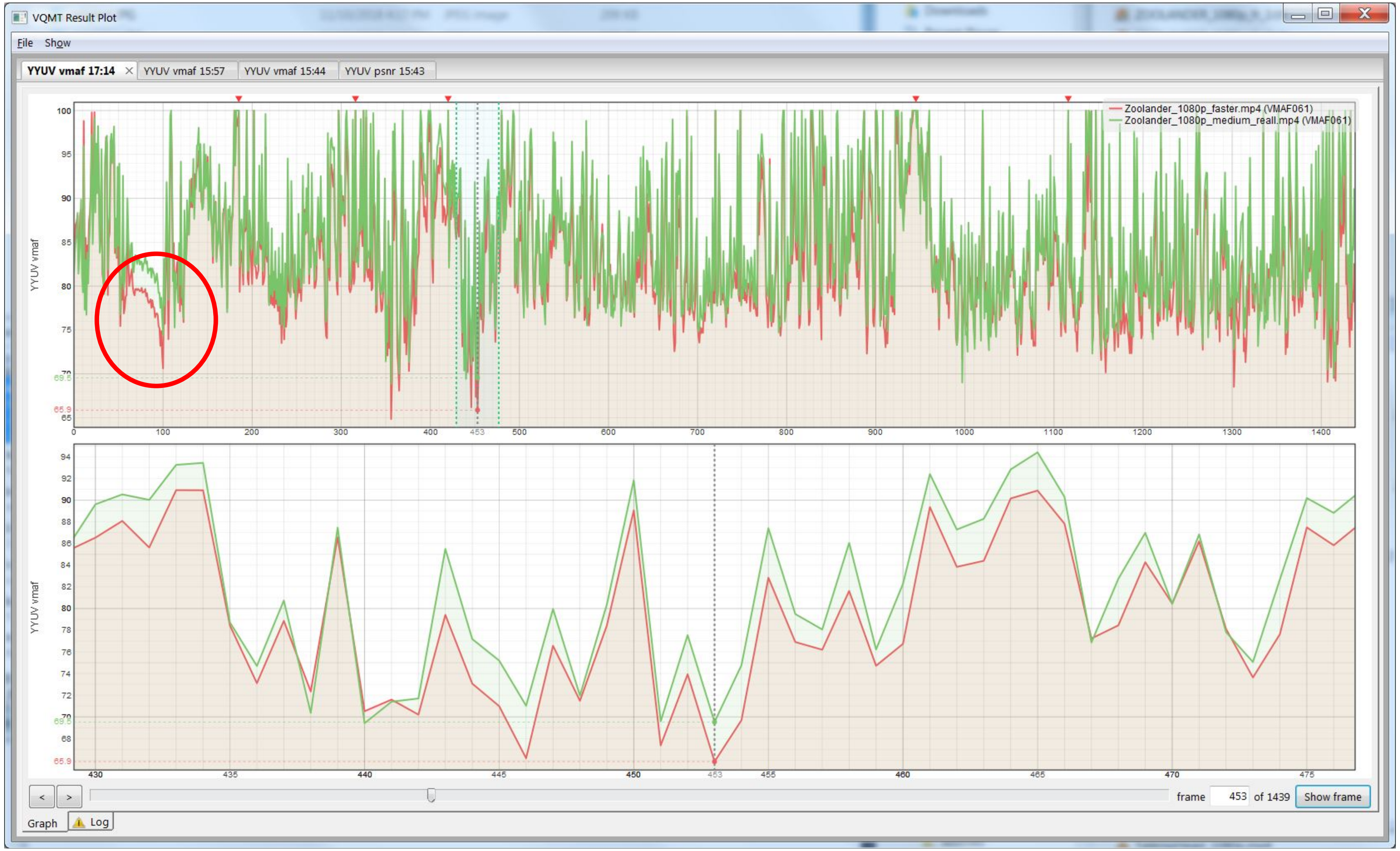
Check Results Plot – Ultrafast (red) vs Medium

- Multiple areas of significant differentiation
- Never use ultrafast (even in live)



Check Results Plot – Faster (red) vs Medium

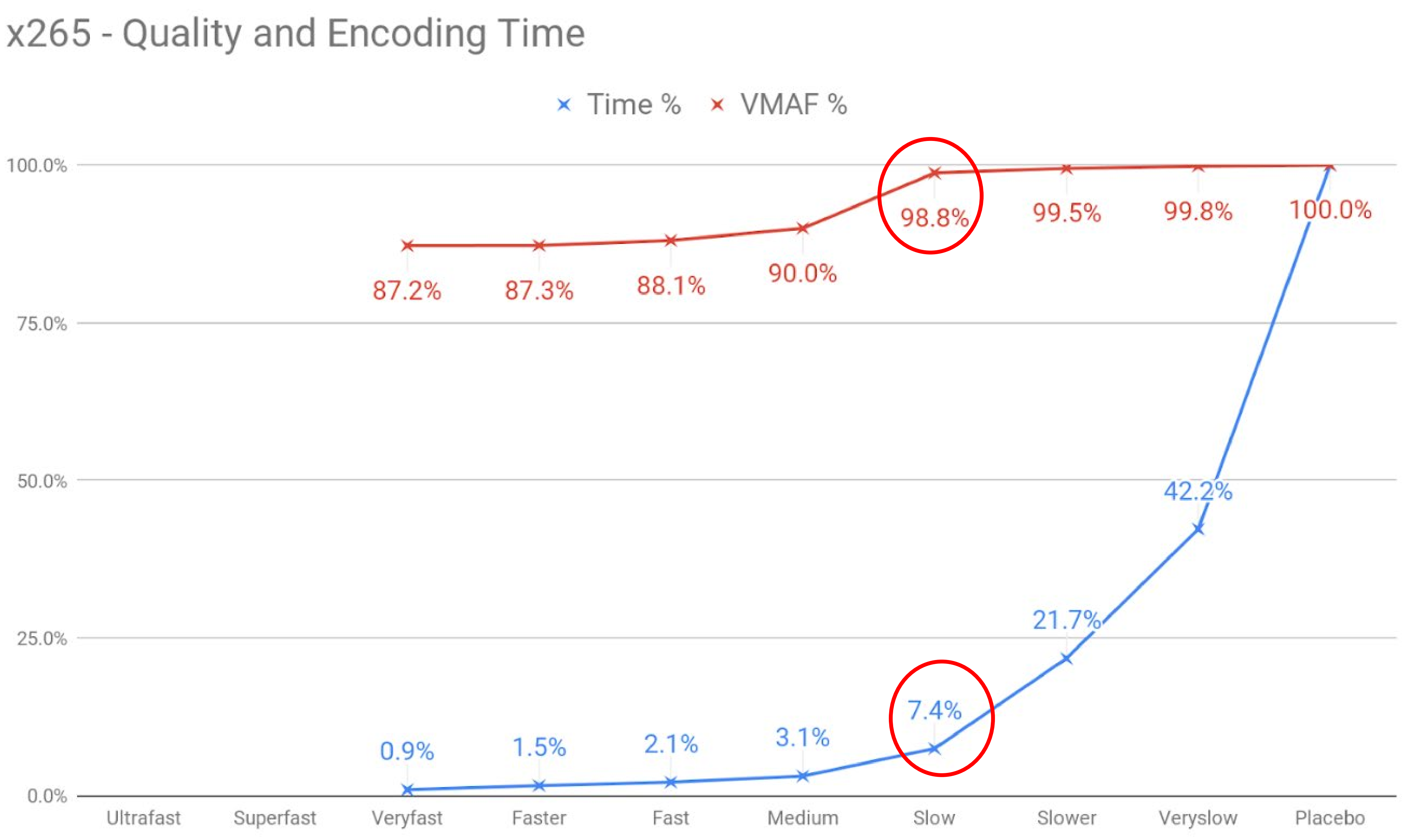
- One problem area, but no major quality differences
- Fast should be acceptable starting point for VOD and live



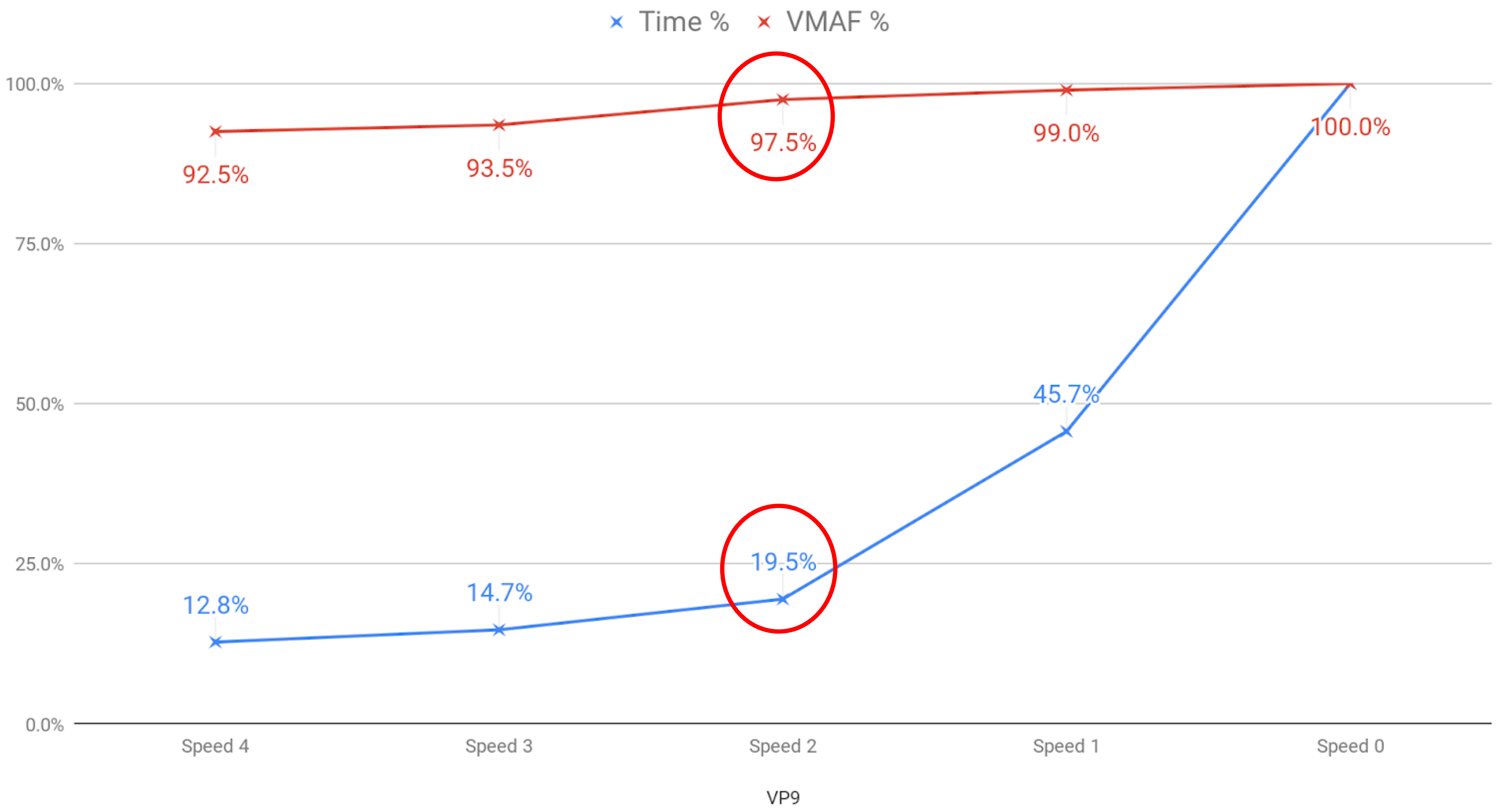
Conclusions

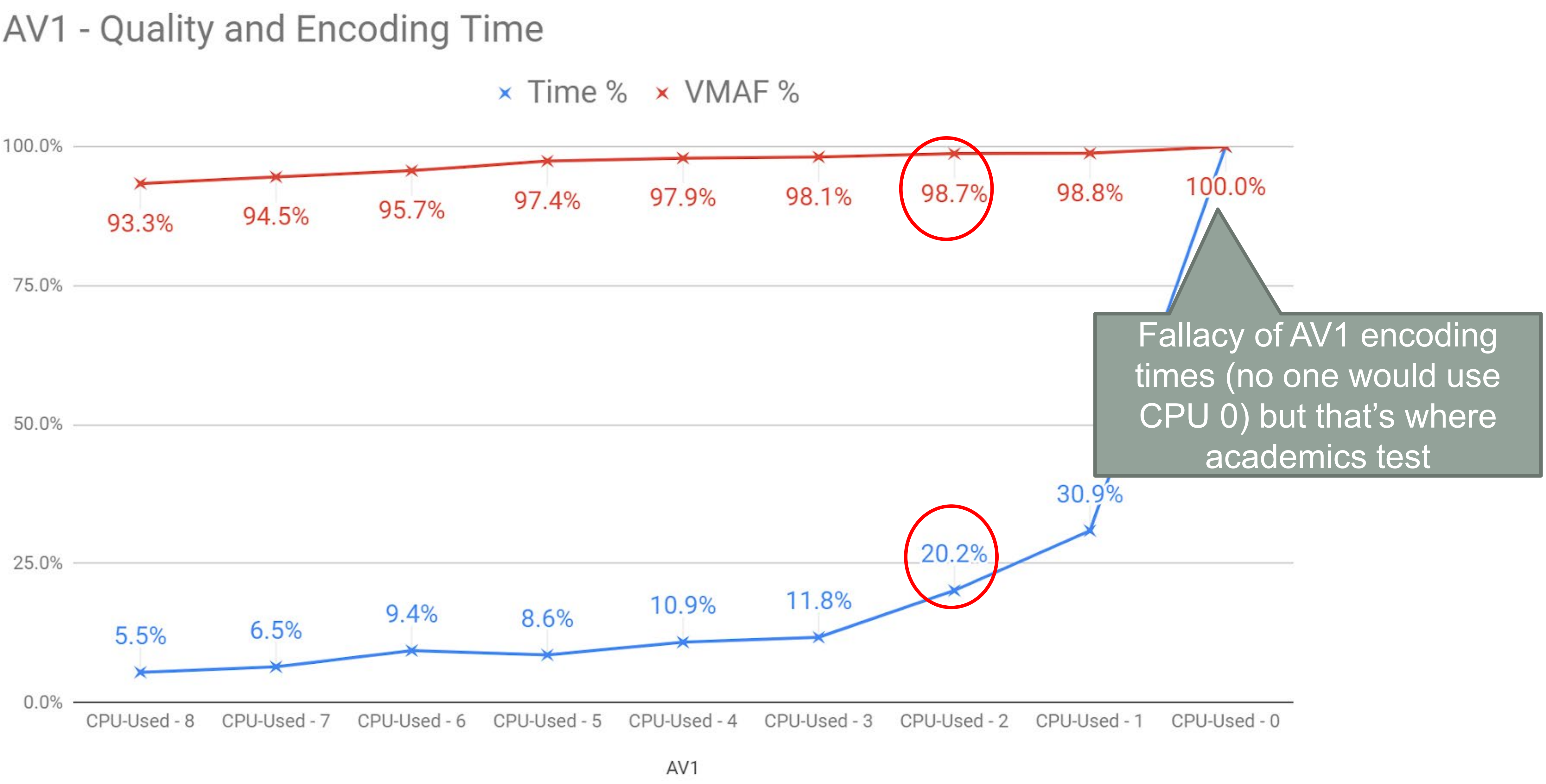
- Faster is best preset for those seeking maximum throughput
- Makes very little sense to go beyond Medium when encoding cost/time is a concern
- Very slow delivers maximum average and low-frame quality; Placebo never seems to make sense

- Single file for recent project



VP9 - Quality and Encoding Time




AV1 - Quality and Encoding Time
Time %
VMAF %
100.0%
75.0%
50.0%
25.0%
0.0%
93.3%
94.5%
95.7%
97.4%
97.9%
98.1%
98.7%
98.8%
100.0%
5.5%
6.5%
9.4%
8.6%
10.9%
11.8%
20.2%
30.9%
CPU-Used - 8
CPU-Used - 7
CPU-Used - 6
CPU-Used - 5
CPU-Used - 4
CPU-Used - 3
CPU-Used - 2
CPU-Used - 1
CPU-Used - 0
AV1
Fallacy of AV1 encoding times (no one would use CPU 0) but that's where academics test

Bottom Line

- Whenever you use a new codec or encoder create a similar model around key quality/encoding time tradeoff
  - Use multiple files
  - Track lowest quality as well as average
  - Make sure transient quality issues (if any) will be noticeable to the viewer

Implementing Per-Title Encoding

- What is it?
  - Identifying the optimal encoding ladder for a single-video file (or category of files)
- Procedure:
  - 1. Find appropriate maximum data rate
  - 2. Choose minimum data rate
  - 3. Fill in rungs between
  - 4. Find optimal resolution for each rung
- How this changes for advanced codecs
- How this changes for different types of content

Finding the Maximum Rung

- Use constant rate factor (CRF) encoding to gauge complexity
- What is CRF
  - An encoding mode in x264, x265, VP9
  - Adjusts data rate to achieve target quality
  - Quality range is 1-51; lower levels are higher quality

```
FFmpeg -i input.mp4  -b:v 5000k output.mp4
```

Delivers 5 Mbps; quality varies

```
FFmpeg -i input.mp4  -crf 23 output.mp4
```

Delivers crf 23 quality; bitrate varies

Finding the Top Rung for 1080p Content

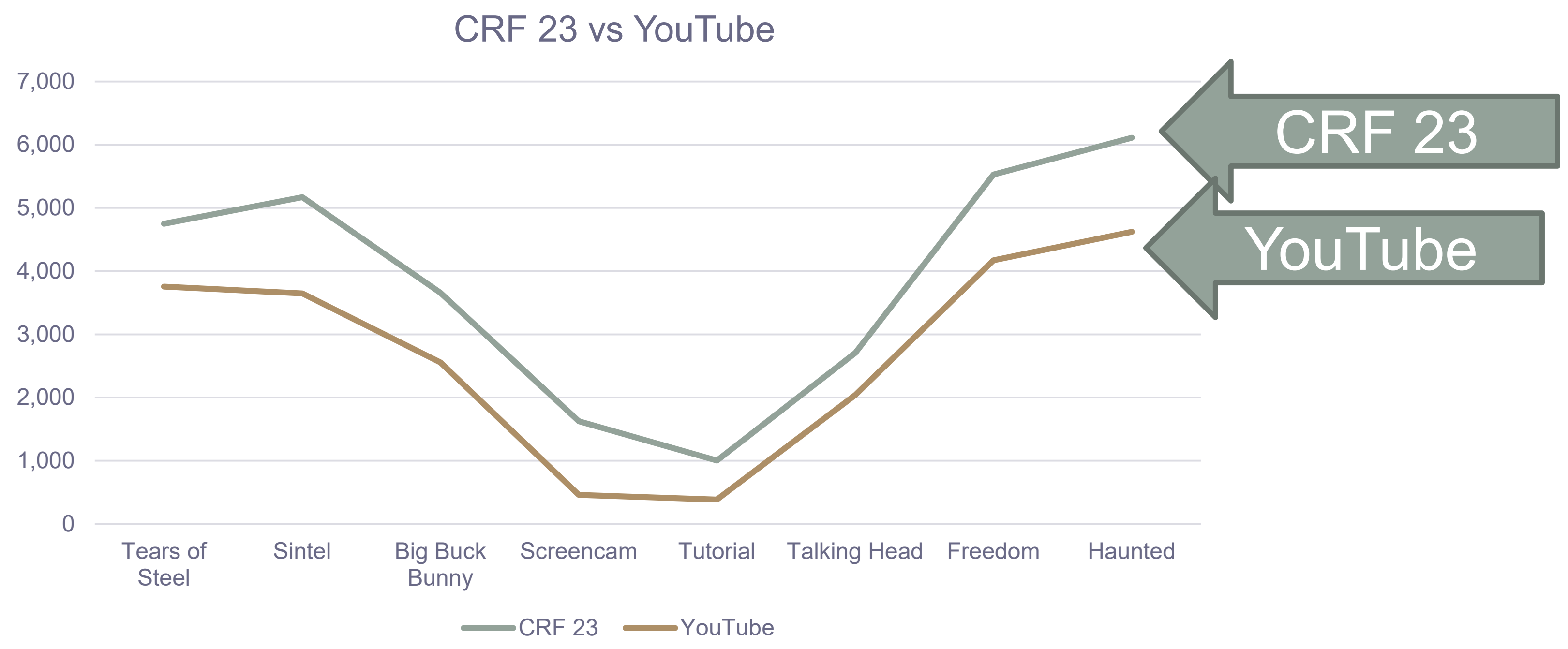
- Compute data rate with CRF 23
  - Encoded 8 files using CRF 23
  - Data rates varied from 1,001 to 6,111 (over 600%)
- Measure VMAF rating
  - Values ranged from 92.74 to 96.88
  - Standard deviation was 1.39 (pretty small)
  - CRF 23 correlates well with VMAF 93
- Analysis
  - At 2.7 Mbps, a talking head video offers same quality as movie at 6.1 Mbps (even lower for synthetic videos)
  - Validating the benefits of per-title encoding

Encoding by the Numbers

CRF23 - 1080p	FPS	Description	Data Rate	VMAF
Tears of Steel	24	Real world/CG movie	4,747	96.45
Sintel	24	Complex animation	5,168	96.96
Big Buck Bunny	30	Simple animation	3,657	96.88
Screencam	30	Camtasia-based video	1,625	96.59
Tutorial	30	PowerPoint and talking head	1,001	96.68
Talking Head	30	Simple talking head	2,706	95.47
Freedom	30	Concert footage	5,527	95.90
Haunted	30	DSLR movie-like production	6,111	92.74
Average			**3,818**	**95.96**
Standard deviation				**1.39**

- Conclusion:
  - CRF 23 with x.264 typically delivers VMAF 93 or higher
  - VMAF 93 is the “magic number,” either no flaws or no irritating flaws

Reality Check: YouTube Comparison



- Upload files to YouTube; measure data rate for H264-encoded files
  - Very popular files now encoded with VP9/AV1 – these are minimum quality
- YouTube uses AI-based per-title optimization
- Pattern very similar

- YouTube averages 1 Mbps lower
- 3 VMAF points lower (1/2 JND)
- More validation that CRF 23 and VMAF 93 predict acceptable quality

Choosing the Data Rate for Individual Rungs

- Step 1: Choose highest – VMAF 93 - 96
- Step 2: Choose lowest – slowest speed you want to serve
- Once you know the highest/lowest add rungs between 1.5 and 2x apart
  - You don't strand viewers at lower quality levels
  - Rungs aren't so close together that you switch needlessly
- Step 3: fill in the blanks (between 150/200% apart)

200 kbps

2x

400 kbps

2x

800 kbps

1.75x

1400 kbps

1. 5x

2100 kbps

1. 5x

3100 kbps

1. 5x

4600 kbps

Encoding Ladder

- We know the data rates
- Next up; resolution

	Data Rate	Resolution
Rung 1	4600	
Rung 2	3100	
Rung 3	2100	
Rung 4	1400	
Rung 5	800	
Rung 6	400	
Rung 7	200	

What Resolution?

- Goal: Find best quality resolution at each data rate
- Derived from Netflix approach
  - Compute VMAF scores at multiple resolutions at each data rate
  - Choose the best quality resolution (green) at each data rate

H.264	1080p	720p	540p	432p	360p	270p	234p
5000	96.22						
4800	96.01						
4600	95.80	95.27					
4400	95.55	95.10					
4200	95.30	94.96					
4000	94.96	94.73					
3800	94.60	94.53					
3600	94.14	94.30					
3400	93.70	93.99					
3200	93.11	93.64					
3000	92.48	93.24					
2800	91.70	92.78					
2600	90.75	92.25					
2400	89.70	91.59	90.39				
2200	88.37	90.80	89.76				
2000	86.72	89.85	88.95	86.93			
1800	84.68	88.66	88.00	86.10			
1600	82.13	87.13	86.77	85.02	81.58		
1400	78.65	85.19	85.16	83.67	80.28		
1200	73.91	82.56	83.01	81.84	78.57		
1000	67.39	78.86	80.02	79.24	76.19		
900	63.18	76.39	77.98	77.47	74.60	66.66	60.58
800	57.93	73.25	75.51	75.34	72.68	65.11	59.23
700	51.47	69.42	72.34	72.59	70.23	63.14	57.49
600	43.12	64.52	68.37	69.11	67.12	60.70	55.33
500	33.31	58.05	63.13	64.66	63.04	57.52	52.46
400	20.82	49.48	56.00	58.46	57.48	53.13	48.59
300	9.74	37.56	45.95	49.62	49.60	46.80	42.96
200	3.73	20.40	30.87	36.12	37.48	36.88	34.03
100		2.75	8.08	14.45	17.50	19.85	18.66

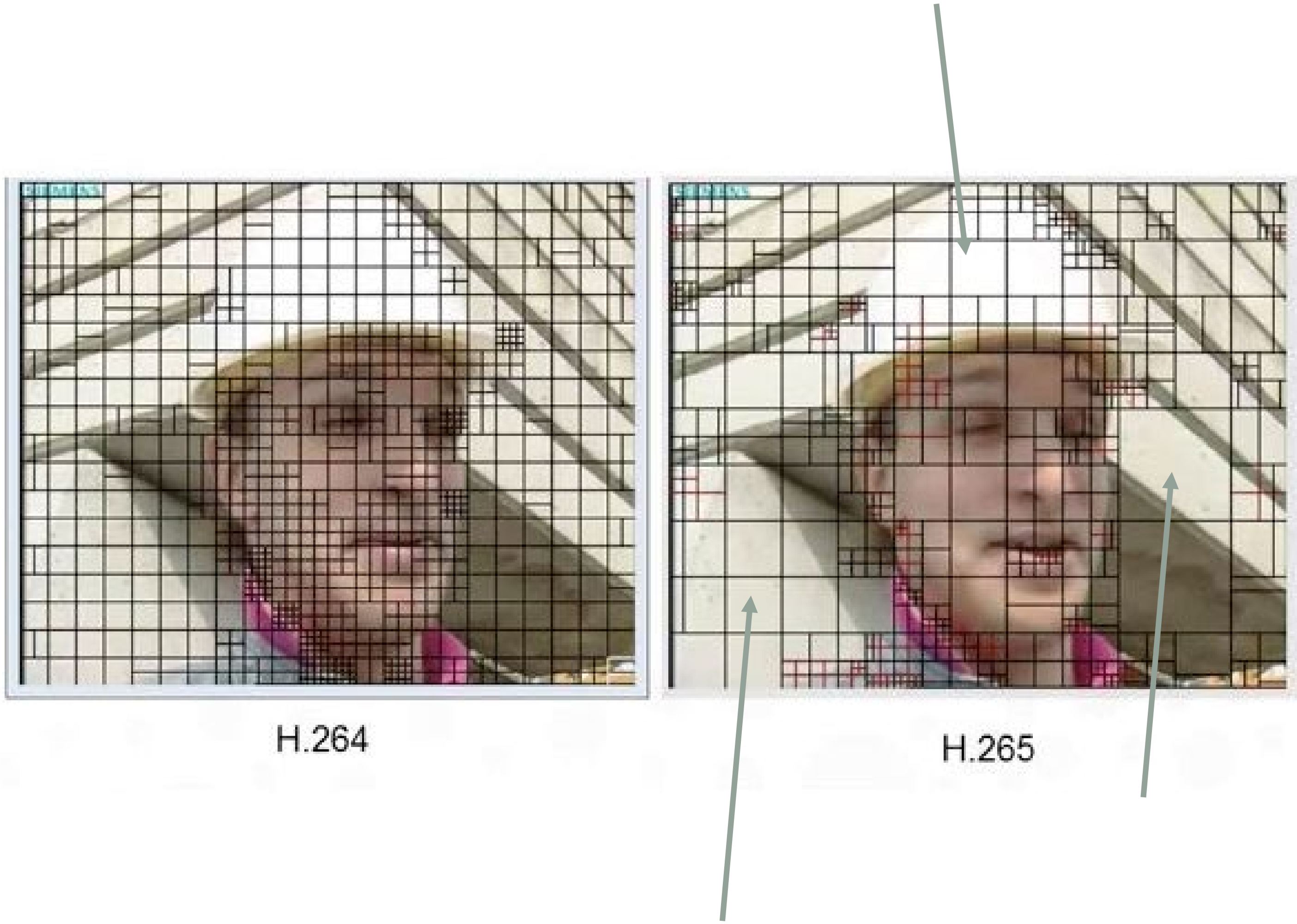
Encoding Ladder

- We know the data rates
- We know the resolutions
- All done

	Data Rate	Resolution
Rung 1	4600	1080p
Rung 2	3100	720p
Rung 3	2100	720p
Rung 4	1400	720p
Rung 5	800	540p
Rung 6	400	432p
Rung 7	200	360p

How Does This Change with Advanced Codecs?

- HEVC (and VP9/AV1) are more efficient
- One prominent advantage – larger block sizes
  - H.264 – 16x16
  - HEVC – 64x64
  - VP9 – 64x64
  - AV1 – 128x128
- Can encode large frame sizes more efficiently than H.264
- Typically translates to better quality at higher resolutions



Proof – Tears of Steel

H.264

H.264	1080p	720p	540p	432p	360p	270p	234p
5000	96.22						
4800	96.01						
4600	95.80	95.27					
4400	95.55	95.10					
4200	95.30	94.96					
4000	94.96	[illegible]					
3800	94.60	94.53					
3600	94.14	94.30					
3400	93.70	93.99					
3200	93.11	93.64					
3000	92.48	93.24					
2800	91.70	92.78					
2600	90.75	92.25					
2400	89.70	91.59	90.39				
2200	88.37	90.80	89.76				
2000	86.72	89.85	88.95	86.93			
1800	84.68	88.66	88.00	86.10			
1600	82.13	87.13	86.77	85.02	81.58		
1400	78.65	85.19	85.16	83.67	80.28		
1200	73.91	82.56	83.01	81.84	78.57		
1000	67.39	78.86	80.02	79.24	76.19		
900	63.18	76.39	77.98	77.47	74.60	66.66	60.58
800	57.93	73.25	75.51	75.34	72.68	65.11	59.23
700	51.47	69.42	72.34	72.59	70.23	63.14	57.49
600	43.12	64.52	68.37	69.11	67.12	60.70	55.33
500	33.31	58.05	63.13	64.66	63.04	57.52	52.46
400	20.82	49.48	56.00	58.46	57.48	53.13	48.59
300	9.74	37.56	45.95	49.62	49.60	46.80	42.96
200	3.73	20.40	30.87	36.12	37.48	36.88	34.03
100		2.75	8.08	14.45	17.50	19.85	18.66

HEVC

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.67						
4800	97.55						
4600	97.44						
4400	97.31						
4200	97.17						
4000	97.01						
3800	96.84						
3600	96.63						
3400	96.41						
3200	[illegible]	95.41					
3000	95.86	95.16					
2800	[illegible]	94.87					
2600	95.09	94.52					
2400	[illegible]	94.12	92.09				
2200	[illegible]	93.63	91.62				
2000	93.16	93.02	91.05	88.30			
1800	92.18	92.25	90.34	87.63			
1600	90.94	91.27	89.44	86.78	83.18		
1400	89.36	89.97	88.27	85.69	82.12		
1200	87.30	88.26	86.68	84.22	80.73		
1000	84.42	85.84	84.46	82.20	78.79		
900	82.39	84.21	83.02	80.86	77.51	68.45	62.18
800	80.03	82.20	81.23	79.19	75.91	67.09	[illegible]
700	77.04	79.67	78.90	77.07	73.91	65.38	[illegible]
600	73.10	76.34	75.88	74.29	71.36	63.21	[illegible]
500	68.11	71.98	71.82	70.61	67.89	60.30	[illegible]
400	61.01	65.92	66.31	65.54	63.19	[illegible]	[illegible]
300	50.13	57.34	58.21	58.06	[illegible]	50.40	[illegible]
200	25.00	44.30	45.88	46.47	45.24	40.96	[illegible]
100	4.14	13.75	24.62	26.16	25.85	23.86	21.53

1080p best quality at far lower data rates than H.264

Lower resolutions don't provide the best quality

Bottom Line: Don't use same encoding ladder for H.264 and advanced codecs

What About Different Types of Content?

- In general:
  - Synthetic videos encode at higher quality at lower bitrates
  - Look better at higher resolutions
    - Push 1080p lower down in the encoding ladder
    - Push 720p further down the ladder
- Not huge difference here, but much more profound for screencams and similar videos
- Compute different ladders for different types of content
  - Particularly synthetic (animation, screencam) vs. real world

Tears of Steel (real world/CG)

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.67						
4800	97.55						
4600	97.44						
4400	97.31						
4200	97.17						
4000	97.01						
3800	96.84						
3600	96.63						
3400	96.41						
3200	96.15	95.41					
3000	95.86	95.16					
2800	95.52	94.87					
2600	95.09	94.52					
2400	94.58	94.12	92.09				
2200	93.97	93.63	91.62				
2000	93.16	93.02	91.05	88.30			
1800	92.18	92.25	[illegible].34	87.63			
1600	90.94	91.27	89.44	86.78	83.18		
1400	89.36	89.97	88.27	85.69	82.12		
1200	87.30	88.26	86.68	84.22	80.73		
1000	84.42	85.84	84.46	82.20	78.79		
900	82.39	84.21	83.02	80.86	77.51	68.45	62.18
800	80.03	82.20	81.23	79.19	75.91	67.09	60.92
700	77.04	79.67	78.90	77.07	73.91	65.38	59.35
600	73.10	76.34	75.88	74.29	71.36	63.21	57.34
500	68.11	71.98	71.62	70.61	67.89	60.30	54.69
400	61.01	65.92	66.31	65.54	63.19	56.29	51.05
300	50.13	57.34	58.21	58.06	56.18	50.40	45.[illegible]
200	25.00	44.30	45.88	46.47	45.24	40.96	37.13
100	4.14	13.75	24.62	26.16	25.85	23.86	21.53

Sintel (animation)

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.83						
4800	97.74						
4600	97.63						
4400	97.50						
4200	97.36						
4000	97.19						
3800	97.01						
3600	96.78						
3400	96.52						
3200	96.22	94.39					
3000	95.86	94.11					
2800	95.45	93.78					
2600	94.94	93.40					
2400	94.32	92.93	89.84				
2200	93.62	92.37	89.34				
2000	92.72	91.69	88.71	85.40			
1800	91.63	90.84	87.94	84.72			
1600	90.21	89.76	87.00	83.84	79.64		
1400	88.44	88.36	[illegible].74	82.74	78.62		
1200	86.02	86.39	84.07	81.24	77.24		
1000	82.81	83.73	81.70	79.13	75.35		
900	80.79	82.02	80.16	77.76	74.10	64.67	58.74
800	78.22	79.83	78.25	76.06	72.55	63.43	57.63
700	75.22	77.22	75.91	73.94	70.64	61.88	56.22
600	71.44	73.84	72.94	71.27	68.17	59.87	54.42
500	66.61	69.68	69.13	67.71	64.90	57.24	52.02
400	60.19	63.98	63.94	62.97	60.47	53.61	48.73
300	48.81	56.19	56.62	56.2[illegible]	54.16	48.26	43.81
200	26.36	44.11	45.66	53.05	44.22	39.79	36.06
100	5.17	15.45	23.86	26.96	26.53	24.50	21.89

Conclusion

- Use different resolutions and switch points for different types of content
  - Particularly synthetic vs. real world videos
    - Synthetic equals animations, screencams, PowerPoint-based videos, CG-based videos

Questions

- Should be: 1:40

Questions

- Should be: 1:40

Implementing Per-Category Encoding

- Now you know how to create an encoding ladder for a single file
- How do you evaluate different categories of content?
- Once you choose the new top rung, use techniques discussed last lesson to create encoding ladder

Implementing Per-Category Encoding

- Scenario
  - Streaming publisher has multiple genres but is using single ladder for all; tuned for acceptable quality for hardest to encode videos (~8 mbps)
- Task
  - Are there genres that could be switched to a lower bitrate ladder (~ 5 mbps) without noticeably degrading QoE?
- Process steps
  - Step 1: Simple triage with CRF 23 – 2-minute segments. Identify genres consistently around 5 Mbps with ~93 VMAF
  - Step 2: Encode at new ladder using normal parameters (2-pass VBR); check file quality against original encode
  - Step 3: View bad frames/regions to determine if typical viewer would notice
  - Step 4: Repeat with full-length clips
  - Step 5: Roll out to limited audience and cross fingers

Step 1 – Triage at CRF 23/21

Channel 1 - General	CRF 23	VMAF	CRF 21	VMAF
Genre 1 Show 1	3,262	92.53	4,561	94.23
Genre 1 Show 2	3,646	93.42	4,977	94.84
Genre 1 Show 3	3,056	89.96	5,009	91.79
Genre 1 Show 4	5,697	94.44	5,397	96.20
Genre 1 Show 5	4,295	94.08	5,869	95.53
Genre 1 Show 6	3,799	92.99	5,966	93.17
Genre 1 Show 7	3,458	89.93	6,015	91.89
Genre 1 Show 8	3,868	88.64	6,584	91.05
Genre 1 Show 9	4,994	94.77	6,641	96.02
Genre 1 Show 10	5,588	93.60	7,518	95.58
Genre 1 Show 11	6,032	93.97	8,056	95.59
Genre 1 Show 12	12,415	90.40	17,251	93.17
Average	**5,009**	**92.39**	**6,987**	**94.09**
Channel 2 - Game shows	**CRF 23**	**VMAF**	**CRF 21**	**VMAF**
Genre 2 Show 1	4,314	92.17	6,183	93.85
Genre 2 Show 2	3,500	93.28	4,941	94.95
Genre 2 Show 3	4,280	92.02	6,189	93.53
Average	**4,031**	**92.49**	**5,771**	**94.11**
Channel 3 - Talk shows	**CRF 23**	**VMAF**	**CRF 21**	**VMAF**
Genre 3 Show 1	3,290	92.81	4,889	94.49
Genre 3 Show 2	3,529	92.60	5,110	94.24
Average	**3,410**	**92.70**	**5,000**	**94.36**
Genre 4 - Sports	**CRF 23**	**VMAF**	**CRF 21**	**VMAF**
Genre 4 Show 1	3,874	93.02	5,678	94.60
Genre 4 Show 2	4,935	93.83	6,880	95.40
Genre 4 Show 3	6,401	94.27	9,180	96.12
Average	**5,070**	**93.71**	**7,246**	**95.37**
Genre 5 - Science	**CRF 23**	**VMAF**	**CRF 21**	**VMAF**
Genre 5 Show 1	2,299	92.25	3,407	93.63
Genre 5 Show 2	4,485	93.08	6,476	94.65
Genre 5 Show 3	3,740	93.30	5,152	94.57
Genre 5 Show 4	4,644	95.41	6,285	96.76
Average	**3,792**	**93.51**	**5,330**	**94.90**

Lots of shows well under 5 mbps

Some much higher / category no good

Good data rate/VMAF OK Good candidate

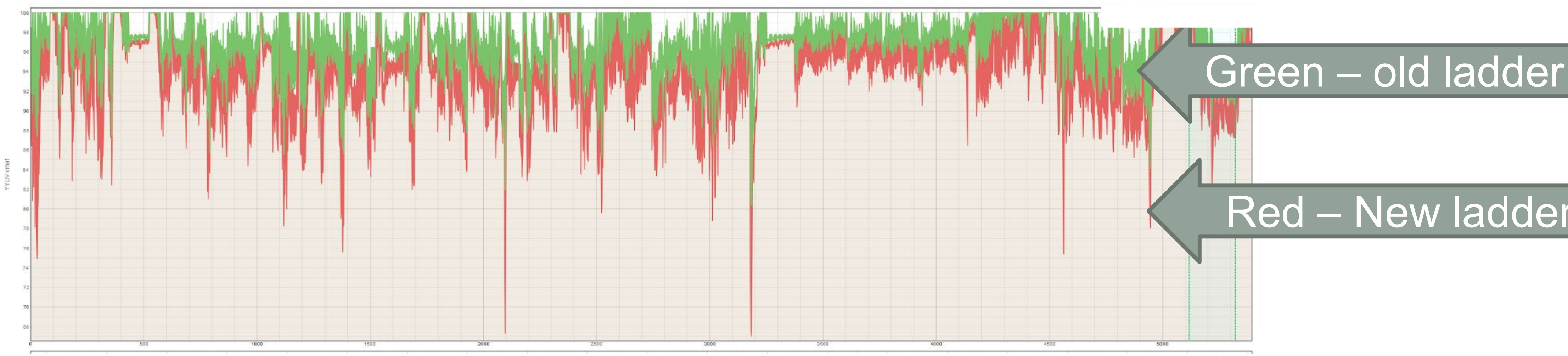
Ditto

Data rate too high

Good data rate/VMAF OK Good candidate

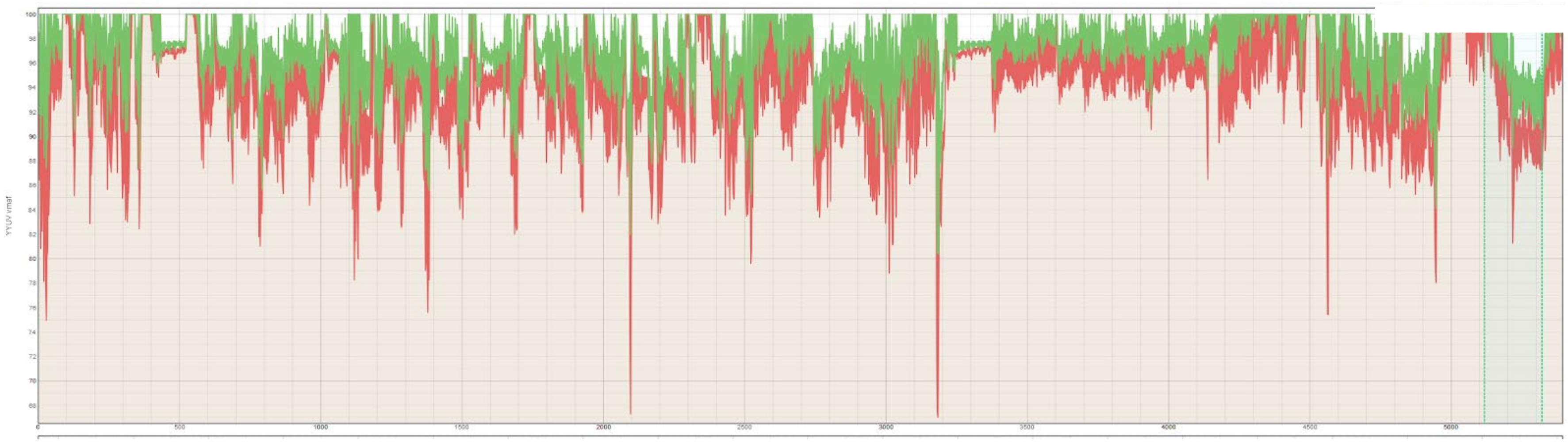
- Wanted same ladder for all shows in the same channel
- Question: Which genres good candidates for 5 mbps max data rate
- Start with 2-minute excerpts
- Gauge complexity with CRF 23 and CRF 21
- Looking for genres with consistent data rates and quality levels

Step 2: Encode with New Ladder/Step 3: Check for Flaws



- Encode 2-minute segments to new target using production encoder/encoding technique
  - CRF gauges complexity
  - Use production encoder (at new target data rate) to compare file against existing encode
- Step 3: Identify problem frames and view them (MSU VQMT/SSIMWAVE tools excellent for this)
  - TOS – 5 mbps – 97.1 VMAF
  - TOS – 8 mbps – 98.4 VMAF
- If quality delta is noticeable, watch the video in real time to determine if typical viewer would notice the difference

Step 4: Once Targets Identified-Repeat with Full-Length Shows



- Full length shows very time-consuming to analyze
- If no major differences, move to step 5

Step 5: Roll Out to Limited Audience

- Roll-out to limited audience
- Gauge reaction
- If no one notices, create the encoding ladder using techniques shown on the last tutorial

What Worked and What Didn't

What worked

- Separate ladder for talk shows, game shows, and sitcoms for major OTT producer
  - Proved that 5 Mbps delivered 93+ VMAF for these types of shows
  - Action shows needed 8 Mbps
- Online training company
  - One ladder for screencam/PowerPoint (2 rungs)
  - One for real world videos (5 rungs)
- Online bike videos
  - Real world needed 1080p to achieve 93 VMAF
  - Simple yoga/stretching videos fine at 720p

What didn't work

- Separate ladders for different kinds of movies (action, drama, comedy etc)
  - Just too much differential within each category
- Separate ladders for animations vs. movies
  - Again, just too much differential – Sintel vs. Big Buck Bunny vs SpongeBob

How Can You Use These Techniques

- What didn’t work
  - Separate ladders for different kinds of movie (action, etc)
    - Just too much differential within each category
  - Separate ladders for animations vs. movies
    - Just too much differential – Sintel vs. Big Buck Bunny vs SpongeBob

Questions

- Should be: 1:40