A Comparison of Structural Similarity Indexes for Adaptive Video Content

Estêvão C. Monteiro and Ricardo E. Scholz, *UFPE*

***Abstract*—In a world of diverse connected devices, with varying bandwidth and video decoding capabilities, along with a billionaire entertainment industry, adaptive video streaming services have become prevalent in the Internet. In such systems, several versions of same content are produced according to targeted devices and bandwidth, while attempting to maintain a threshold of quality against human perception. Modeling human perception of images has been a prolific research field for years, and the SSIM index has gained wide adoption and has been tuned and extended with varying techniques. As well, the x264 encoder for H.264 video has been established as the most efficient publicly available video encoder, notorious for its psychovisual tunings. This paper presents an extensible, open-source framework for numerous SSIM-based tools designed for high performance and Full HD support, called jVQA. Upon this framework, the performances of 32 combinations of 10 SSIM techniques are compared and it is found that gradient correlation greatly outperforms variance correlation in measuring x264's psychovisual improvements, while computing faster, among many more specific findings, including original improvements proposed in this paper.**

***Index Terms*—Avisynth, error analysis, H.264, image quality, Java, perceptual image coding, performance evaluation, SSIM, streaming media, structural similarity, video compression, x264.**

I. Introduction

In a world of diverse connected devices, with varying bandwidth and video decoding capabilities, along with a billionaire entertainment industry, adaptive video streaming services have become prevalent in the Internet. In such systems, several versions of same content are produced according to targeted devices and bandwidth, while attempting to maintain a threshold of quality against human perception [10]. Modeling human perception of images has been a prolific research field for years, and the SSIM index has gained wide adoption and has been tuned and extended with varying techniques [1] [2] [6] [9] [13] [14]. As well, the x264 encoder for H.264 video has been established as the most efficient publicly available video encoder, notorious for its psychovisual tunings [7] [8]. This paper presents an extensible, open-source framework for numerous SSIM-based tools designed for high performance and Full HD support, called jVQA - Video Quality Assessment in Java. Upon this framework, the performances of 32 combinations of 10 SSIM techniques are compared in an attempt to objectively measure the otherwise subjective tunings of x264.

The key concept in comparing video versions is video fidelity. Video is a sequence of frames, which are digitized as color points called pixels. The most basic concept of video fidelity in digital medium comes from the amount of information or bits used to represent each pixel: bits per pixel (per frame). Since raw, uncompressed video produces an enormous volume of highly redundant data in all dimensions, in practice digital video is always employed in compressed form. Particularly, Web-based streaming requires high compression, which must be efficient to avoid or manage fidelity loss [10]. Video compression is a sophisticated science and art that exploits both convenient mathematical transformations and peculiarities of the human visual system (HVS). For this reason, bits per pixel alone are not a useful metric, although it is always part of the equation.

A reference software for jVQA is the Moscow State University Video Quality Measurement Tool [3], which computes precise and fast versions of SSIM, MS-SSIM and 3-SSIM, among other metrics. This tool uses Avisynth input, which provides a useful abstraction of the video decoding function [11]. Avisynth input is also convenient for encoding with x264. Thus, it has also been found a convenient input for jVQA.

Ideally, jVQA would be implemented in the C++ language for dynamic linking with Avisynth. Due to time constraints and lack of experience with C++, however, the authors chose to implement in Java. Since Java runs on a virtual machine - JVM, it was necessary to implement a native access component, which was called JNAVI - Java Native Access for Avisynth.

II. Managing Avisynth Input With JNAVI

Avisynth is a video frame server for Windows operating systems that relies on either the system's native decoders (Windows DirectShow) or specific plugins for decoding video (such as an ffmpeg wrapper). It consumes simple scripts for instructions and produces decoded frames. A dynamic link library is provided that encapsulates all frame-serving functions. Through this library, the frame byte stream becomes available to client applications.

This work was developed under orientation of Prof. Tsang Ing-Ren, at Pernambuco Federal University of Brazil – UFPE as a requirement to the Image Processing graduate course in June, 2014.

E. C. Monteiro, B.Sc., is with the Data Processing Federal Service of Brazil – SERPRO of Brazil (estevao at gmail dot com) and M.Sc. student at UFPE.

R. E. Scholz, M.Sc., is with the 6th Region Regional Labour of Brazil – TRT6 (reps at cin dot ufpe dot br) and Ph.D student at UFPE.

The Java Native Access library was employed by JNAVI for accessing the Avisynth DLL's functions [12]. Thus, Avisynth was encapsulated and abstracted, and convenience functions were implemented. For example, MPEG-based video is based on the planar, 12-bit Y'UV12 colorspace. This colorspace separates the luminance plane from the two chrominance planes (differentials between blue and yellow and green and red) and subsamples chrominance to ¼ pixels so the original 24 bits per pixel are halved, based upon the premise that the HVS is less sensitive to chroma variations than luma. JNAVI implements convenience functions for obtaining the byte streams of each plane separately.

JNAVI is published as free, open source software (FOSS) at SourceForge.net [5].

III. JVQA Design

SSIM is an image metric, not a video metric *per se*. While there are video metrics such as MOVIE and spatio-temporal SSIM [16], these are not yet implemented in jVQA. Since a single minute of video at typical 24 frames per second make for 1440 frames, it is important that metrics computation be efficient. Also, high-definition content has recently become commonplace, jumping from trivial 300,000 pixels in 480p per frame to 2 million in 1080p, and Ultra-HD 4K is starting to be marketed. Since metrics are done over the 8-bit luminance component of decoded frames, each pixel equals to one byte, which can inflate memory usage rapidly if left unmanaged. Unfortunately, this has found to presently be the case with JNAVI. Java does not allow for memory management by the programmer, so it will be necessary to encapsulate Avisynth's DLL with a custom DLL to correctly free memory pointers as soon as they are no longer needed.

In order to maintain maximum control over image processing efficiency, the authors chose to forgo any image processing library and implement each image operation from scratch: correlations, convolutions, downsampling, box filtering, Gaussian filtering and Roberts and Sobel gradients, for instance. Along with native memory concerns, it is also important to not overexert the JVM's memory by using adequate primitives. Ideally this would be "byte", but in Java all integer types are signed, in this case ranging from -128 to 127, so inefficient conversions would be needed for the required arithmetic operations. For this reason, the 16-bit integer "short" type is preferred over "byte". It is also important to avoid floating-point operations, substituting with integers whenever possible. Also, when convolving matrices, it is most efficient to apply as many combined operations as possible at each pixel scan.

Finally, sensible object-oriented design was employed, not only for component organization and avoid code redundancy, but also to allow for smooth recombination of elements as well as reuse and extension. Along with JNAVI, jVQA is published as FOSS at SourceForge.net [4, 5].

IV. Implemented Metrics

Initially, jVQA implements the SSIM index, as well as some of its variations. The original SSIM index proposed by Wang and Bovik analyses images' luminance plane over means (luminance), variances (contrast) and correlation of variances (structure), employing an 8x8 box filter to achieve local isotropy, producing real values theoretically in the range -1 (completely different) to 1 (identical) [18]. Later formalization and improvement with Sheikh and Simoncelli used stabilization of zeros in divisions by adding fixed constants and substituted the box filter for an 11x11 Gaussian 1.5 filter [13]. A multiscale approach for contrast and structure, MS-SSIM, has also been proposed by those authors, which pools the responses of full scale and four successive ¼ downscalings, while computing luminance only for the smallest scale; this was shown to better approximate subjective mean opinion scores [14]. In the SSIM official web site, an ideal single scale is also proposed, by downsampling by a factor that approximates the smallest frame dimensions to 256 [17]; this has also been implemented in jVQA.

$$
\begin{aligned}
l(\mathbf{x},\mathbf{y}) &= \frac{2\,\mu_x\,\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},\\
c(\mathbf{x},\mathbf{y}) &= \frac{2\,\sigma_x\,\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},\\
s(\mathbf{x},\mathbf{y}) &= \frac{\sigma_{xy} + C_3}{\sigma_x\,\sigma_y + C_3},\\
C_1 &= (K_1\,L)^2\,,\; C_2 = (K_2\,L)^2\\
&\text{and } C_3 = C_2/2
\end{aligned}
\tag{1}
$$

$$
\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}. \tag{2}
$$

$$
\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l_M(\mathbf{x},\mathbf{y})]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(\mathbf{x},\mathbf{y})]^{\beta_j} [s_j(\mathbf{x},\mathbf{y})]^{\gamma_j} \tag{3}
$$

Independent authors Chen, Yang & Xie proposed to substitute Sobel gradient for variance in SSIM, thus G-SSIM [6]. Later, this was found useful by Chen & Bovik in their optimizations for Fast SSIM (F-SSIM), where they have shown that Roberts gradient is sufficiently accurate and much faster than Sobel for the purposes of SSIM [9]. Also, while Chen, Yang & Xie computed magnitude simply by the sum of responses of the pair of Sobel filters, Bovik proposes to sum ¼ of the lowest response to the full highest response instead, which helps reduce dynamic range overflow. Regarding luminance, since Rouse & Hemami proposed that the luminance index usually contributes nothing to the index [2] (unless an actual specific luminance distortion occurs), Chen & Bovik propose to dispense with the Gaussian filter for this index, instead using a fast box filter and the integral image technique for computing means. The present paper proposes further to subsample the luminance computation by a factor equal to the box filter's side, which is 8 in this case, since all the information of the 64 pixels is condensed by the box filter into the final, blurred pixel, thus further reducing the cost of computing the lumi-

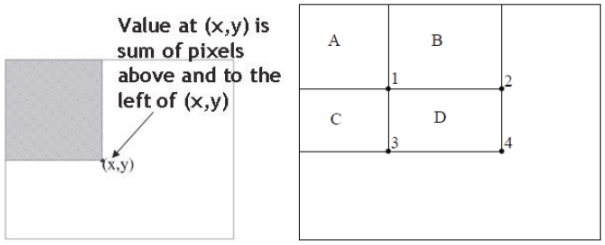

Fig. 1. Left: Integral image. Right: How to compute sum value over region D in integral image domain.

0	0	0	1	1	0	0	0
0	0	1	2	2	1	0	0
0	1	2	4	4	2	1	0
1	2	4	8	8	4	2	1
1	2	4	8	8	4	2	1
0	1	2	4	4	2	1	0
0	0	1	2	2	1	0	0
0	0	0	1	1	0	0	0

Fig. 2. 8x8 integer approximation of Gaussian window.

nance index, in a manner similar to MS-SSIM. Finally, Chen & Bovik propose to avoid entirely any floating point operations besides computing the final index, by substituting an integral approximation of the Gaussian filter and limiting its size to 8x8 instead of 11x11.

In a later paper, Rouse & Hemami propose to dispense with stabilizer constants, instead applying conditional treatment to avoid division by zero, as if instead of zero the value was an infinitely small number [1]. This should produce a wider amplitude of values, since in SSIM the constant C2 for a dynamic range of 0-255 amounts to 58.5225, which, although operating with squares, is not an insignificant value and almost negates intensities close to that value, especially intensities less than 8. The wider amplitude is of great interest for comparison between images that are very close in quality, as in the case of adaptive streaming and psychovisual tuning, for otherwise, the index responses are too close for significant differentiation. In jVQA, however, Rouse & Hemami's conditional treatment has been modified in response to anomalies during experimentation and for greater performance and is presented here as a new proposal. Thus, for the luminance index:

```
if mx = my then index = 1
//treats zero divisor and resolves fast
else if mx = 0 or my = 0, then index = 0
//resolves fast
else increment both mx and mx and compute standard
division without stabilizer constants.
```

And, for the consolidated contrast x structure index:

```
sx = sy, then index = 1
//treats zero divisor and resolves fast
else if sxy = 0 or sx = 0 or sy = 0, then index = 0
//resolves fast
else compute standard division without stabilizer
constants.
```

The modified luminance index increments values because otherwise the index resulting of 0 and 1 would be 0, while the index resulting of 1 and 2 is 0.8, which would cause a wide distortion. Thus, dynamic range in index computation is shifted from 0-255 to 1-256, a minor adjustment that stabilizes the index's results.

Preliminary testing indicates that a constant stabilizer of 1 produces the same information as 58.5225, only with a wider dynamic range of intensities. While this should be the case for any positive, non-zero constant, when normalizing the error map to a dynamic range of 0-255, a stabilizer of 0.0001 negates many useful values by approximating to 0. Thus, for normalized error map visualization purposes, this paper proposes that a constant stabilizer 1 is probably more useful than both 0 and the standard SSIM stabilizer. Regardless of visualization, since they widen index amplitude, both 0 and 1 seem more useful for comparing images that are known beforehand to be highly similar.

V. Psychovisual Ranking

Every lossy video encoder faces an important decision: which information to keep and which to discard. Mean squared error is the widest and longest adopted decision metric for this quantization, in the form of peak signal-to-noise ratio (PSNR). However, it has long been proven to be a poor metric, favoring blurriness over fine detail and completely disregarding image structure [19]. X264 can be tuned to maximize PSNR, SSIM or its own psychovisual model. This model is based on calibration of adaptive quantization, visual energy retention, Trellis quantization and in-loop deblocking filter [20] [21]. Adaptive quantization allows for some transform blocks in a frame to be allotted more bits than others and is useful to preserve smoothness in gradients. Visual energy retention favors retention of fine detail, which contributes to human perception of quality. Trellis quantization accounts for entropy coding, i.e., the actual cost of bits, when deciding quantization, which otherwise would be a future step that does not affect quantization decisions. Finally, in-loop deblocking filter is a very useful feature of modern video formats but which must be calibrated to avoid unwanted loss of fine detail.

PSNR and, to a lesser extent, SSIM have been found to favor blurriness, and for this reason rank psychovisual tunings poorly. Thus, when evaluating by traditional, variance-based SSIM, low-bitrate videos tuned by SSIM, PSNR and psychovisual model ("psy") will rank SSIM first and psy last. This paper is interested in a metric that would rank psy first and PSNR last, building upon the SSIM framework. Thus, four classifications are defined: target (psy > SSIM > PSNR), expected (SSIM > PSNR > psy), inverse (PSNR > SSIM > psy) and unexpected (any other ranking).

Since SSIM employs a blurring filter (either box or Gaussian), it is no surprise that such index exhibits a bias towards blurring. This paper will investigate the effect of such filter over psychovisual ranking and index amplitude by comparing results with a single-pixel window.

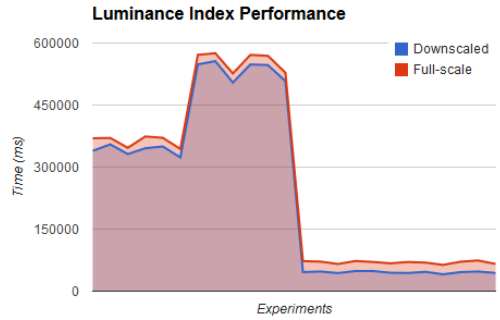

Fig. 3. Luminance index computation performance comparison between full-scale and downscaled approach proposed in this paper. The improvement in performance occurs for all full-scale (downscaling strategy) experiments.

VI. Experimentation Methodology

Experimentation was conducted in order to measure psychovisual ranking adherence, index amplitude and computing performance between the various SSIM-based tools of interest. 5 components or dimensions are compared: structure scale, luminance precision, structure statistic, window type, and division stabilization strategy.

For structure scale, there are 3 options: single full-scale, single down-scaled to 256 and multiscaled. For luminance index precision, also 3 options: Gaussian filtering, full-scale box filtering and downscaled box filtering. For structure statistic, 5 options: variance, Roberts gradient with simple magnitude sum, Roberts gradient with Bovik's magnitude sum, and the Sobel gradient equivalents. For structure window type, there are 4 options: 11x11 precise 1.5 Gaussian, 8x8 average, 8x8 Gaussian integer approximation and single pixel (unfiltered). Finally, for zeros stabilization, there are 2 options: Wang et al's constants or conditional handling adapted from Rouse & Hemami. All these produce 360 combinations.

Some options are sufficiently redundant that may be discarded in the interest of brevity. Insuficient documentation and implementations on the single down-scaled method were identified for comparison, so this is not tested. Since it has previously been estabilished that luminance index is of little contribution, there is insufficient necessity to test precise (Gaussian) luminance. As for statistics, preliminary tests indicate that Sobel and Roberts produce most the same information in the context of correlation and comparation and Roberts is faster, so Sobel is discarded. It is also out of the focus of this paper to compare Bovik's magnitude to simple magnitude, so we are left comparing variance to Robert's gradient by Bovik's magnitudes. Although MSU VQMT provides the option for box filtering as a performance compromise, this paper is more interested in comparing Gauss with unfiltered. Particular interest also resides in how a single pixel window compares to a Gaussian window, as results may be more precise and favor visual energy retention. Since performance is also a concern, the integer approximation is preffered over precise Gaussian.

360 combinations are thus reduced to 32 more significant combinations, composed of: full-scale vs. multi-scale structure, full-scale vs. downscaled box-filtered luminance, variance structure vs. Roberts gradient structure, 8x8 Gaussian window vs. single pixel window, and constant vs. conditional stabilization.

TABLE I
Combinations of Approaches Tested

Dimension	Approach 1	Approach 2
Structure scale	Single, full	Full +4 downscales
Luminance precision	Full-scale average	Downscaled average*
Structure metric	Variance	Roberts gradient
Pooling window	8x8 Gaussian	Single pixel*
Division stabilization	Constant	Conditional*

Approaches proposed in this paper are marked with [*].

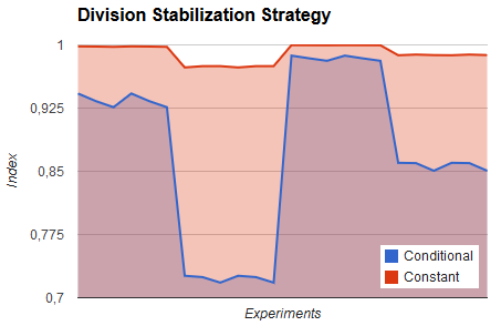

Fig. 4. Division stabilization strategy comparison between constant an conditional approach proposed in this paper. Graphic shows the increase in amplitude values, keeping the proportion between original values.

Since experiments measured execution time, the full batch was run in the same hardware. Two full batches were run in two different hardware: an Intel i7 with 4 GB RAM running Windows 8 and an Intel i5 with same RAM running Windows 7, both with Java Runtime 7. Computation time has been measured strictly over metric computation, separate from file system operations.

Experimentation was conducted over a 1'36” long excerpt of a commercial Blu-ray title with a mix of high and low movement between frames, with a total of 2300 frames. Due to memory leakage in Avisynth, it was necessary to downsample the frames from 1920x1080 to 720x404 and regularly skip frames while batch testing, effectively downsampling framerate by a factor of 2, for the i7 CPU (1150 total frames), and 3, for the i5 CPU (767 total frames). 3 degraded versions were produced in x264 for each of the target tunings: psy (grain), SSIM and PSNR. The degraded versions are all 2-pass 600 kbit/s constrained to a keyframe interval of 96, all IDR-frames, no B-frame pyramid, minimum quantizer = 4, and a 1-second, 900 kbit/s video buffer [7] [10]. With 32 metrics and 3 degraded videos, 96 tests were conducted over 1150 and 767 frames each.

VII. Experimental Results

17 out of 32 metrics were found to adhere to the target psychovisual ranking; 5 resulted in the expected ranking, 4 resulted in the inverse of the target ranking, and 6 resulted in unexpected rankings. However, after testing, a bug was identified in the implementation of unfiltered variance metrics, which

contributed 4 times to target and 4 times to unexpected rankings and may change to other rankings after correction.

Notably, gradient structure always adheres to the target ranking except when simultaneously unfiltered and stabilized by constants. It also produces the widest amplitudes. Naturally, variance tends to favor SSIM tuning, although it produces mixed results for the second and third places in rank. Overall, multiscaling attenuates all parameters' effects and all amplitudes, except for conditional unfiltered gradient.

The single pixel window drastically improves computing time for all combinations, up to 10 times, virtually negating performance variations of other parameters; all results are within 10% and the difference between gradient and variance becomes statistically null (less than 1%). Gradient structure index results become inconsistent between conditional and constant stabilization. Variance structure index results seem to stabilize and negate the effects of other parameters, but amplitude is so extremely reduced (5th decimal for full-scale, 7th for multiscaled) it becomes of little usefulness. Unfiltered gradient with conditional stabilization adheres to the target ranking and might be the best metric of all over all parameters, but since it is inconsistent with constant stabilization and this is not well understood, it requires further testing.

With the approximated Gaussian window, multiscaled processing takes 25% longer than full-scale. This is less than the expected 33%, most likely due to luminance processing time being included. Variance is 50% slower than gradient, also including luminance time. Future testing should disregard luminance time in order to properly compare the performance of the different statistics.

Conditional stabilization improved index amplitude for gradient but reduced it for variance. It also greatly reduces the index for gradient, lowering from 0.9 to 0.8 with Gauss or 0.6 unfiltered; this may be useful but requires further testing. Constant vs. conditional stabilization only affects ranking for variance with Gaussian and gradient unfiltered; all other cases see no change in ranking within other parameters. Conditional stabilization remains a useful tool but requires attention. Surprisingly, no statistical difference in performance was observed.

Downscaled luminance index is consistent with fullscaled up to the 5th decimal, a promising result, except for multiscaled variance with Gaussian, which is anomalous by this and other parameters. With fullscaled structure, it performs around 20 ms per frame faster than fullscaled, which for 767 frames amounts to 12 less seconds, on average. On the other hand, this contribution is diluted in multiscaled structure, which already downscales luminance by default and also computes structure 5 times. In any case, MS-SSIM corroborates the strategy of downsampling luminance and this has proven effective. Also, results confirm the small contribution of luminance to the overall index.

Large anomalies resulted from both Gaussian and unfiltered multiscaled variance with full-scale luminance and conditional stabilization. These probably indicate bugs in the authors' implementation. Since full-scale luminance was found to be reasonably redundant with downscaled luminance and the downscaled luminance versions did not reflect such anomalies, these are not considered significant results.

TABLE II
BEST PERFORMANCES PER GROUP

Group	Best Run Time
Gaussian full-scale gradient	5'42"
Gaussian full-scale variance	8'26"
Gaussian multiscaled gradient	7'10"
Gaussian multiscaled variance	10'44"
Unfiltered full-scale gradient	1'39"
Unfiltered full-scale variance	1'40"
Unfiltered multiscaled gradient	1'48"
Unfiltered multiscaled variance	1'48"

For analysis purposes, results were grouped by structure scale, structure statistic and window type, which are the most significant parameters. Within each group, 4 results are listed combining full or downscaled luminance and conditional or constant stabilization. Index and performance results are mostly consistent within the groups, except for anomalies in unfiltered gradient (both full and multiscaled) and Gaussian multiscaled variance. Performance results for the best of each group are given in Table II. All metrics were also tested for a result of 1 when comparing identical content and passed.

VIII. CONCLUSION

Free, open-source, reusable, object-oriented software was developed with Java and published at SourceForge.net for accessing frames served by Avisynth and computing SSIM-based video quality metrics, with several general-purpose image processing tools. SSIM, Gradient SSIM, Fast SSIM and previously proposed optimizations were implemented in such a way that their individual elements can be combined to compose many different metrics. The most significant 32 combinations of these elements were twice tested against psychovisual ranking, index amplitude and computing performance for 3 different tunings of the x264 encoder. Gradient structure was found to better correlate with the x264's psychovisual techniques than variance structure, which is consistent with the detail retention and gradient optimization techniques employed by the encoder; it also shows greater index amplitude and performs better than variance structure. Downsampling the luminance component of the index proved reliable, while not pooling local statistics with a Gaussian window seems unreliable at this point, requiring deeper investigation. Conditional stabilization of the index showed inconclusive results as well.

This work allowed the authors a deeper understanding of many image processing operations: planar color space processing, convolution, correlation, statistical variance analysis, gradients, box filters, Gaussian filters, image averaging by integral, and performance optimization, among others. It has also been an exercise in statistics and software engineering. The authors were able not only to implement several techniques proposed in six academic papers, but also to propose new and effective approaches.

IX. Future Work

Many tasks remain to make jVQA feasible for practical use by the public:

- memory management on the native side;
- performance test for 1920x1080 content;
- multi-threaded processing;
- correction of two anomalous metric combinations; and
- finishing the graphical user interface.

Experimental results beg deeper investigation into several metrics:

- unfiltered gradients; and
- Gaussian multiscaled variance.

8x8 windows overlap with 8x8 spatial prediction blocks of MPEG video. This is likely not a problem with Gaussian windows, which weight more at the center, but can be a problem with average box filters such as for the luminance index. Avoiding such overlap, such as by the 11x11 window proposed by Wang et al, might improve the indexes' perceptual effectiveness. Since this is only an issue for luminance, which uses a box filter and is downscaled, a 10x10 or 12x12 window would probably be more effective. As well, it would be interesting to compare the Gaussian integer approximation to a simple averaging window, such as implemented in MSU VQMT's fast metrics, coupled with downsampling, as has been proven effective for luminance; such approach would greatly improve performance at the cost of some precision, which might perhaps be reasonably subtle as to preserve the metric's effectiveness.

It is also important to confirm all ranking results against content of varying bit rates. Finally, more metrics deserve implementation in jVQA, such as:

- 3-Component SSIM [15];
- Spatio-temporal SSIM [16]; and
- PSNR.

References

[1] D. Rouse; and S. Hemami, "Analyzing the role of visual structure in the recognition of natural image content with multi-scale SSIM," *SPIE Human Vision and Electronic Imaging XIII*, Feb. 2008.

[2] D. Rouse; and S. Hemami, "Understanding and simplifying the structural similarity metric," in *IEEE International Conference in Image Processing*, pp. 1188-1191, Oct, 2008.

[3] D. Vatolin et al. (2013.) *Moscow State University Video Quality Measurement Tool* [Online]. Available: http://compression.ru/video/ quality_measure/video_measurement_tool_en.html

[4] E. Monteiro. (2014, April). *Java Native Access for Avisynth* [Online]. Available: http://sourceforge.net/projects/jnavi

[5] E. Monteiro and R. Scholz. (2014, May). *Video Quality Assessment in Java* [Online]. Available: http://sourceforge.net/ projects/jvqa

[6] G. Chen; C. Yang; and S. Xie, "Gradient-based structural similarity for image quality assessment," in *IEEE International Conference in Image Processing*, pp. 2929-2932, Oct. 2006.

[7] J. R. C. Patterson. (2012, April). *Video encoding settings for H.264 excellence* [Online]. Available: http://www.lighterra.com/papers/ videoencodingh264

[8] L. Merrit, "X264: a high performance H.264/AVC encoder," *Washington University*, 2006.

[9] M. Chen; and A. C. Bovik, "Fast structural similarity index algorithm," *Journal of Real-Time Image Processing*, vol. 6, ed. 4, pp. 281-287, Dec. 2011.

[10] M. Levkov, "Video encoding and transcoding recommendations for HTTP Dynamic Streaming on the Flash Platform: preliminary recommendations for video on demand," *Adobe Systems,* Oct, 2010.

[11] R. Berg et al. (2012-2014). Avisynth [Online]. Available: http://avisynth.org

[12] T. Wall. (2008). *Java Native Access (JNA)* [Online]. Available: http://github.com/twall/jna.

[13] Z. Wang; A. C. Bovik; H. R. Sheikh; and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no. 4. pp. 600-612, Apr. 2014.

[14] Z. Wang; E. P. Simoncelli; and A. C. Bovik, "Multi-scale structural similarity for image quality assessment", in *IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1398-1402, Nov. 2013.

[15] C. Li and A. C. Bovik, "Content-weighted video quality assessment using a three-component image model", *Journal of Electronic Imaging* 19, issue 1, Jan.-March 2010.

[16] A. K. Moorthy and A. C. Bovik, "Efficient Motion Weighted Spatio-Temporal Video SSIM Index", in *Proc. SPIE 7527, Human Vision and Electronic Imaging XV*, 75271I, Feb. 2010.

[17] Z. Wang et al. *The SSIM Index for Image Quality Assessment* [Online]. Available: http://www.cns.nyu.edu/lcv/ssim

[18] Z. Wang, and A. C. Bovik, "A universal image quality index", *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81-84, March 2002.

[19] Z. Wang and A. C. Bovik, "Mean squared error: love it or leave it? - A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98-117, Jan. 2009.

[20] J. Garret-Glaser. (2008, March 31). *X264 Adaptive Quantization and You* [Online]. Available: http://forums.animesuki.com/showthread.php?t=64485

[21] J. Garret-Glaser. (2008, May 31). *Psy RDO: Official testing thread* [Online]. Available: http://forum.doom9.org/showthread. php?t=138293

APPENDIX A - Experimental results for 12 frames per second (page 1 of 2)

Gaussian fullscale gradient:	**Psy-grain**	**Time**	**SSIM**	**Time**	**PSNR**	**Time**	**Amplitude**	**Ranking**
Conditional, downsampled luma	0,94235214	5'39"432	0,93334317	5'55"077	0,92606817	5'31"559	0,01628397	Target
Conditional, full luma	0,94235129	6'09"849	0,93334201	6'10"496	0,92606698	5'46"786	0,01628431	Target
Constant, downsampled luma	0,99824107	5'45"697	0,99805673	5'49"915	0,99770879	5'23"471	0,00053227	Target
Constant, full luma	0,99823996	6'14"067	0,99805528	6'10"893	0,99770726	5'44"017	0,00053270	Target
Gaussian fullscale variance:								
Conditional, downsampled luma	0,97869169	9'08"479	0,97985273	9'16"300	0,97929336	8'24"402	0,00116103	Inverse
Conditional, full luma	0,97869082	9'31"582	0,97985153	9'35"360	0,97929211	8'46"547	0,00116071	Inverse
Constant, downsampled luma	0,99013999	9'08"309	0,99122574	9'06"804	0,99126759	8'28"034	0,00112760	Expected
Constant, full luma	0,99013889	9'31"472	0,99122431	9'29"180	0,99126606	8'48"429	0,00112718	Expected
Gaussian multiscaled gradient:								
Conditional, downsampled luma	0,98728226	7'33"018	0,98406214	7'36"668	0,98111436	7'07"429	0,00616790	Target
Conditional, full luma	0,98728144	7'40"452	0,98406136	7'01"921	0,98111332	7'16"012	0,00616812	Target
Constant, downsampled luma	0,99973350	7'39"143	0,99967390	7'39"884	0,99958277	7'11"707	0,00015073	Target
Constant, full luma	0,99973255	7'43"001	0,99967296	7'08"997	0,99958154	7'11"798	0,00015100	Target
Gaussian multiscaled variance:								
Conditional, downsampled luma	0,99154924	12'00"702	0,99194642	11'51"672	0,99144383	11'23"311	0,00050259	Unexpected
Conditional, full luma	**0,99999593**	**1'00"065**	**0,99999549**	**0'53"311**	**0,99999494**	**0'54"722**	**0,00000099**	Target
Constant, downsampled luma	0,99654260	11'46"876	0,99687375	11'35"912	0,99672227	11'18"476	0,00033115	Expected
Constant, full luma	0,99154842	12'02"739	0,99194563	11'11"327	0,99144279	11'20"092	0,00050284	Unexpected

Bold-faced results were found incorrectly computed.

APPENDIX A - Experimental results for 12 frames per second (page 2 of 2)

Unfiltered fullscale gradient:	**Psy-grain**	**Time**	**SSIM**	**Time**	**PSNR**	**Time**	**Amplitude**	**Ranking**
Conditional, downsampled luma	0,72604727	0'46"174	0,72446875	0'47"556	0,71793424	0'43"787	0,00811303	Target
Conditional, full luma	0,72604661	1'12"616	0,72446785	1'11"548	0,71793331	1'05"699	0,00811330	Target
Constant, downsampled luma	0,97325227	0'48"686	0,97478602	0'48"680	0,97481230	0'44"553	0,00156003	Inverse
Constant, full luma	0,97325119	1'13"040	0,97478460	1'10"598	0,97481080	1'07"363	0,00155961	Inverse
Unfiltered fullscale variance:								
Conditional, downsampled luma	**0,99988178**	**0'43"995**	**0,99988898**	**0'46"764**	**0,99987266**	**0'40"780**	**0,00001632**	**Unexpected**
Conditional, full luma	**0,99988088**	**1'10"534**	**0,99988776**	**1'09"069**	**0,99987139**	**1'03"543**	**0,00001637**	**Unexpected**
Constant, downsampled luma	**0,99987496**	**0'46"153**	**0,99988259**	**0'47"528**	**0,99986514**	**0'44"117**	**0,00001745**	**Unexpected**
Constant, full luma	**0,99987385**	**1'11"224**	**0,99988114**	**1'14"232**	**0,99986360**	**1'05"818**	**0,00001753**	**Unexpected**
Unfiltered multiscaled gradient:								
Conditional, downsampled luma	0,86009529	1'03"543	0,85973426	1'01"943	0,85065132	0'58"394	0,00944396	Target
Conditional, full luma	0,86009457	1'03"748	0,85973358	0'56"476	0,85065044	0'58"456	0,00944414	Target
Constant, downsampled luma	0,98772358	1'06"147	0,98852156	1'05"713	0,98791329	1'00"164	0,00079797	Expected
Constant, full luma	0,98772264	1'05"668	0,98852063	0'57"357	0,98791209	1'02"181	0,00079799	Expected
Unfiltered multiscaled variance:								
Conditional, downsampled luma	**0,99999706**	**0'54"847**	**0,99999663**	**0'57"342**	**0,99999638**	**0'51"978**	**0,00000067**	**Target**
Conditional, full luma	**0,99973255**	**7'45"240**	**0,99967296**	**6'59"419**	**0,99958154**	**7'11"474**	**0,00015100**	**Target**
Constant, downsampled luma	**0,99999688**	**1'00"404**	**0,99999642**	**0'59"364**	**0,99999617**	**0'55"205**	**0,00000071**	**Target**
Constant, full luma	**0,99999623**	**0'55"620**	**0,99999584**	**0'52"820**	**0,99999533**	**0'53"063**	**0,00000090**	**Target**

Bold-faced results were found incorrectly computed.