MSU Video Quality Measurement Tool (VQMT)

Documentation

PRO & Premium

version 12.0 BETA

Nov 2019
CS MSU Graphics & Media Lab
Video Group
http://www.compression.ru/video/

Contents

Overview

Brief Description

MSU Video Quality Measurement Tool is professional software that is used to perform deep comparative objective analysis of video quality. The main functionality of this software is to calculate objective quality metrics for digital multimedia content (video or image) using reference (when comparing one or several processed/compressed/distorted video sequences to original one) or non-reference (when analyzing content and getting mark of its quality) types of analysis.

The main application areas for this tool are:

- **Video/picture codec quality analyzing** (developers, quality assessment experts or even users can perform comparative quality analysis using some other codecs as reference, or to compare different releases of the one codec, etc.)
    - Moscow State University, CS MSU Graphics&Media Lab performs many codec analysis using this tool, some of them you can find here: http://www.compression.ru/video/codec_comparison/index_en.html
    - See also x264 Codec Capabilities analysis from YUVsoft Corp. http://yuvsoft.com/pdf/x264_parameters_comparison.pdf
- **Video/image processing algorithm comparison** (when analyzing objective quality of different processing algorithms). See Video Denoiser Comparison (Comparison of Different Noise Reduction and Removal Solutions) from YUVsoft Corp for example: http://yuvsoft.com/pdf/video_denoiser_comparison.html
- **Different algorithms for image and video analysis** (when analyzing unknown algorithm from other company to understand which pixels of the video or image it changes and how strong this change is)
- **And many other tasks**

MSU Quality Measurement Tool uses more than 13 objective quality metrics, has special SDK for objective metrics development and implementation, handles with different video formats and color spaces. Also it has options to visualize objective metrics value with internal visualizator for every metrics (it is internal option of each metric) and other useful features for complete, fast and accurate objective quality assessment for multimedia content.

"PRO version" is a special edition for IT companies. It helps them to carry out massive comparisons of their technologies using batch processing of big numbers of files and flexible options for measurements and has no technical limitations.

"PREMIUM version" is a special license, that doesn't limit installations.

Change Log

[!] – Known bug

[+] – New Feature

[*] – Other

12.0

[+] Online metric calculation

[+] Visualization for specific frames

[+] Computed metric values in Preview

[*] VQMT will not stop if some files, but not all, are done

[+] Automatic color selection in command line possible

[*] Color specification in command line changed

[*] Now colors named by single letter, i. e. Y instead if YYUV

[+] PSNR metric now supports single value for RGB and YUV

[*] Single PSNR metric instead of 4

[+] Processing of multiple colors simultaneously possible in GUI

[+] Preview window: zoom

[+] Preview window: timestamp & frame

[+] Preview window: no freezing when changing frame

[+] Preview window: pixel indicator & pixel information

[+] Preview window: individual pixel values

[+] Preview window: clickable miniview

[+] Preview window: hotkeys improved

[+] Preview window: vertical comparison mode

[+] Preview window: slider mode

[+] Preview window: information in fullscreen mode

[+] NIQE: visualization

[*] NIQE will output common mean and NIQE mean as separate accumulated values

[+] More average values: statistical data, specific metric values

[*] New JSON format: column-vise JSON values

[+] Bitrates in JSON

[*] New CSV format

[*] Legacy mode to bring VQMT closer to VQMT 11

[+] Subsampling mode: skipping frames to improve performance

[+] Performance settings in GUI, new performance setting – metric parallelism

[+] Sample conversion setting

[+] More RGB ↔ YUV tables

[*] Changed value range in the following metrics: Delta, MSE, MSAD

[*] Option “-subsampling” renamed to “-no-upscale-uv”

[+] Considerably improved precision for SSIM Fast and MSSSIM fast metrics

[+] Corrected precision in SSIM-CUDA, SSIM-OpenCL, Blocking

[+] Improved multi-core support

[+] Some metrics optimized

[+] VQMT internally optimized

[+] Results plot window optimized

[+] Completely refreshed VQMT log output

[+] Information in console window: FPS, estimated time, etc.

[+] Average values in console output

[+] Additional information in log: average FPS, exit status, etc

[+] More accurate FPS during calculation

[+] Consider correct sample range in all metrics

[+] New bad frames features

11.2

[+] CentOs Linux now is supported

[+] Added FFmpeg scaling algorithms

[*] Fixed PSNR average value

[*] Fixed OpenCL metrics (SSIM, 3SSIM, MSSSIM)

[*] Fixed incorrect values in bi_psnr metric

[*] Fixed BFM metric

[*] Fixed NIQE metric

[*] 3SSIM-CUDA bug fixed

[*] Error reporting in CUDA metrics, accuracity in CUDA metric fixed for compatibility with OpenCL, GPU_Id metrics

[*] CSV fixes - correct escaping, line endings

[*] RGB48 pixel format support fix

[*] Other bugfixes

11.1

[+] Comparison on different resolution (geometry correction)

[+] New metrics (SI, TI)

[+] Saving bitrates

[!] And bugfixes

11.0

[+] Support for 1400+ new RAW formats

[+] Support for standard names for RAW formats

[+] Support new RGB ↔ YUV tables

[+] Support HDR videos and more HDR image formats: 3 more times video formats now can be lossless

[+] Result table with metric values inside VQMT

[+] Automatic command-line generation from GUI

[+] Plot of bitrate

[+] Actual VMAF support: modern models, 4k models, confidence intervals

[+] New blurring metric

[+] New Noise Estimation metric

[+] Linux command line considerably improved

[+] Better performance on some cases

[+] New quality control standards

[+] Measurement on subsampled U and V

[+] Latest CUDA engine

[!] And bugfixes

[!] Only x64-bit edition from this version

10.2

[+] PREMIUM licensing introduced

[!] And bugfixes

10.1

[+] Support of perspective and modern VMAF metric. Many settings guarantee full metric support.

[+] Now it's possible to determine quality even without a reference via new NIQE metric.

[+] Improved usability: Drag & Drop files and changing of file order by single click.

[+] The -stdin parameter allows you to transmit data directly from the output of another program, such as FFmpeg, without saving multi-gigabyte files. Acceleration and simplification (PRO and DEMO only)

[+] New formats with support for HDR: extended support for TIFF and PXM family. (HDR only in PRO and DEMO)

[+] Better usability of command line - a lot of improvements while maintaining full backward compatibility (PRO and DEMO only)

[+] Built-in profiler. It will show which operations take the most time and allow you to accurately measure the performance of VQMT (PRO and DEMO only)

[+] Better JSON output

[+] TIFF as format for bad frames

[+] More functions in Linux version: more metrics, plugins support, OpenCL support (PRO and DEMO only)

[!] And bugfixes

[!] Now we recommend specifying original input using new key '-orig' instead of traditional '-in'

10.0 BETA

[+] New look of main window

[+] Saving and loading VQMT projects

[+] Status bar with diagnostic in main window

[+] Report generation

[+] Saving result plot in arbitrary image formats

[+] New clear VQMT folder structure

[+] Improved using vqmt console in PRO version

[+] Command line Linux utility

9.1

[*] Fixed incorrect results while using second processed video

[+] Added legend to result plot

[+] Saving log from GUI

[*] Fixed copying plot to clipboard

[*] Using *.YUV bug fixed: first frame could be duplicated

[*] Improved h265 support

[*] Index file building bug fixed

9.0 BETA

[+] New look of result window

[+] Using interface while calculation is processing and while viewing results

[+] Viewing multiple results at the same time

[+] Switch between calculation log and results plot

[+] Extended information about progress: count of processed frames, FPS, elapsed time, estimated time

[+] Pretty and configurable result plot

[+] Saving CSV after calculation finished

[+] Saving JSON result file in GUI

[+] Saving results' plot as SVG file

8.1

[+] Added checking for updates.

[+] Using single image as source do not truncate all other inputs to one frame.

[*] Improved video previewing, fixed opening visualization video inside VQMT.

[*] Fixed crashes when use plugin metric.

[*] Fixed possible incorrect negative PSNR.

[*] Fixed mask behavior.

[+] Recognize patterns 720p etc. in the name of RAW file to detect resolution automatically.

[+] Hotkeys in Preview and Fullscreen window considerably improved.

[+] Allow to change video source and frame inside Preview window.

[*] Bugfixes in Preview window performed.

[*] Fixed green frame in some types of file, incorrect reading of some color spaces.

8.0 BETA

[+] Added new visualization methods: Lossless Video, Lossy Video and TIFF files.

[+] Preview window replaced.

[+] Full-screen preview, display selection.

[+] Side-by-side preview.

[+] Inspecting visualization inside VQMT.

7.1

[*] Crashes fixed.

[*] Improved command line output in PRO version.

[*] Preview display fix.

[*] Seeking and offsetting YUV files fixed.

7.0 BETA

[+] Added support for OpenCL device interface. Efficient calculation on different devices.

[+] Speedup of PSNR and SSIM metrics.

[+] Similar metrics joined.

6.2

[+] Added support for JSON output in console.

[*] Incorrect results with RGB metric on RGB video fixed.

6.1 BETA

[*] Fixed bug: hangs in console while using files in network directories.

[*] Fixed incorrect behavior of .Y4M files in GUI.

[*] Fixed crashes when the length of all files was not able to be detected.

6.0 BETA

[+] VQMT became about 3 times faster.

[+] 16 and 32 bpp floating point TIFF files supported (PRO version).

[+] Added “-threads” command line argument to PRO version, which allows to control CPU usage.

[*] Advanced mask processing - allowed shifts from exact black color for indication black area.

[*] Fixed crashes and incorrect results while using mask.

5.2

[*] Fixed bug: hangs in console while using files in network directories.

[*] Files incorrect behaviour of .Y4M files in GUI.

5.1

[+] Support comparing ranges of input videos.

[*] Fixed crashed while using YUV files with more than 8bpp.

[*] Corrected RGB <-> YUV conversion procedures.

[*] Fixed incorrect transformation for some input formats.

[*] 10, 14 and 16 bpp RGB format plain order changed to R, G, B.

5.0 BETA

[+] Added Open File Wizard to help with opening and previewing files;

[+] Use wizard or file picker for selection of input file;

[+] Wizard supports drag&drop mechanism;

[+] Now user can select mode for opening file (FFmpeg, AviFile, automatic AviSynth, RAW file, image or image sequence and others);

[+] Special mode to compare the results of opening a file using different modes;

[+] Automatic selection of the best open mode for specified file;

[+] More settings for customization opening process;

[+] Added automatic generation of video index file that guarantees file will be opened and correct previewed;

[+] Number of files available for opening and previewing increased;

[+] Added support for images as input files: *.jpg, *.png, *.tif, more formats of *.bmp and many others;

[+] Added support for using image sequences as input video;

[+] Previewing raw files (*.yuv, etc.) "on the fly";

[*] PRO version console interface is more user friendly with full compatibility to previous version;

4.4

[+] Number of files available for metric calculation increased;

[+] The number of supported devices for running CUDA metrics increased;

[*] Unable to run metric for single file bug fixed;

[*] Unable to view results if some minor error occurred bug fixed;

[*] CUDA metrics crash fixed;

4.3 BETA

[+] Number of files available for metric calculation increased. Now metric can be calculated for the files that not available for preview;

[+] Speed up of file opening in metric calculation process for some types of files;

[+] Standard VQMT plug-ins is now supported;

[*] Memory leak fixed in VQMT.

4.2 BETA

[+] Added native support for *.mkv, *.flv and some other containers and codecs;

[*] Stability fixes.

4.1 BETA

[+] Added FFmpeg file reading support, the number of supported formats greatly increased. Using AviSynth is not recommended;

[*] Fixed x64 version crashed;

[*] Stability fixes.

3.2

[*] Existence of AviSynth determining fixed;

[+] AviSynth for VQMT as standalone installer;

[*] AviSynth plug-in opening fixed;

[*] Fixed unsuccessful file opening in AviSynth mode;

[+] AviSynth mode now supported in console; [+] All dependencies now are immediately in installer, no more redistributable packages needed;

[+] Main menu and desktop labels fixed to determine Pro, Free and Pro Demo license;

[+] Executable file metadata errors fixed;

[+] Cosmetic fixed in Interface and file naming: revision number added to version naming;

[*] Fixed crashes in 64-bit version on multiple platforms;

[*] Fixed crashes and hangs after: the press of Process button, viewing of analyses result, other events.

3.1

[+] Changed to CUDA 5.0 toolkit, added Kepler support (Compute Capability 3.0)

[*] Stability fixes

[*] Fixed major bug with masking

3.0

[+] Added stSSIM metric.

[+] Added ".y4m" raw video internal support

[+] Added Autoupdate feature for free version (our PRO customers receive updates automatically)

[+] Added CUDA realization for SSIM-based metrics (SSIM, 3-SSIM, MS-SSIM. Requires CUDA-capable device)

[*] Added subjective comparison for the most popular metrics (see metrics info)

[*] Added 64-bit version of MSU VQMT

[!] Program crashes due memory lack when -metr ALL specified with large (i.e. 1280x720) video frames.

2.7.3

[*] Fixed bug with MSSIM metric causing source frame change.

[*] Fixed some metrics inaccuracy causing different metric values by enabling\disabling visualization.

[!] Program crashes due memory lack when -metr ALL specified with large (i.e. 1280x720) video frames.

2.7.2

[*] Fixed bug causing incorrect metric values, when using 3SSIM and MSSSIM

[*] Fixed bug causing incorrect PSNR metric values in CSV files

[*] Fixed bug causing no metric calculation for large (>4gb) files

[*] Not existing directory specified in "-cod" parameter will be created now and processing will not cancel.

[!] Program crashes due memory lack when -metr ALL specified with large (i.e. 1280x720) video frames.

2.7.1

[*] Fixed bug in CVS file generation. Sometimes first frame metric value was empty.

[*] Fixed bug causing incorrect MSE metric values after calculating SSIM metric.

2.7

[+] MSSSIM (fast and precise) metric implemented.

[+] 3SSIM metric implemented.

[*] Fixed bug in calculation of VQM metric under Windows 7.

[*] Fixed bug during program launch on some computers.

2.6 (Windows Vista & Windows 7 support)

[*] Fixed bug in Scene Change Detection plugin when working under Windows Vista or Windows 7.

[*] Fixed bug in saving visualization video when running on Windows Vista or Windows 7.

[*] Fixed dependency with vcomp.dll

2.5

[*] Fixed bug in processing of *.YUV files with non-standard resolution.

[*] Fixed bug in loading the mask from *.YUV files.

[*] Fixed bug in masking of L (LUV colorspace) component.

[*] Fixed bug in processing of non-standard resolution *.AVS files.

[+] Video with any resolution is now supported by all metrics. Video with resolution which is not appropriate for some metric is now expanded (via data duplication, separately for each metric) to make resolution acceptable.

[+] 1.95 times speed up of command line tool multiple metrics calculation on average (PRO version only)

[+] YUV files with size more than 2Gb are supported now

[*] Fixed bug in calculation of SSIM (precise) for second reference file

[*] Fixed bug in conversion from RGB32 to YUV color spaces for video with non-standard resolutions (affects calculation of metric for *.AVI files)

[+] Output directory for *.CSV and visualization files is automatically set to folder of last specified reference file

2.01 beta

[+] 1.5 times speed up of command line tool multiple metrics calculation on average (PRO version only)

[+] Masking is added

[*] Fixed bug in 4:2:2 raw files with more than 8 bits per component support

2.0 beta

[+] *.MOV, *.VOB, *.WMV, *.MP4, *.MPG, *.MKV, *.FLV formats support simplified
[+] HDTV support (PRO version only)
[+] Raw files with more than 8 bits color depth per component are supported (PRO version only)
[+] Alternative SSIM and PSNR are added for compatibility with other implementations.
[+] New version of *.CSV files with average metric values (PRO version only)
[+] Minor acceleration
[+] Preview buttons are added
[+] Options save is improved
[+] All MSU plugins are renamed (names are now more correct in GUI and simpler to call from PRO console)
[*] MSU Noise Estimation plugin bug with incorrect (identical) values for some videos is fixed.
[*] MSU Noise Estimation and MSU BI-PSNR plugins provide correct information about their home pages now.
[*] MSU BI-PSNR plugin crash during visualizing a metric for video sequences with dimensions less than 255 is fixed

1.52

[*] Error in saving CSV file for comparative analysis fixed

1.51

[*] Error at the opening YUV-files fixed

[+] YUV10, YUV16 formats for YUV files added

[+] Improved codecs support

1.5

[*] Set of interface fixes

1.4

[*] Bug fixing in BMP processing (visualization saving, etc)

1.3

[*] Bug with YV12 yuv files fixed

1.2

[+] Now it is possible to compress visualization

[+] Plug-in mechanism released

[!] Problem with some DV codecs

1.0

[+] More YUV file types are supported, including YV12, YUY2, YUV

[+] Supports Unicode

[+] Visualization dialog was extensively reworked

[*] Interface is more user-friendly

0.81

[+] New AVI Reading system (support large AVI Files, VP 70)

[*] Bug fixing in final dialog

0.8

[+] New dialog with visualization of the comparison and for comparison of the selected frames added.

0.75

[+] Improved codecs support (x264)

[+] Now data from YUV-AVI files is extracted without conversion

[*] Bug fixing: #NAN in VQM calculation, calculation of RGB-metrics for YUV-files

[*] MSU Blocking Metric changed

0.74

[+] MSE, MSAD metrics added

[+] Saving of mean value of the metric added

[+] Improved codecs support (DivX3, WMV, mjpeg2000)

[*] Delta metric was changed

[*] Bug fixing (AviSynth - wrong result for comparison of three files)

0.73

[+] All color spaces from AviSynth are supported

[+] I420 (IYUV) support added

[+] XviD and B-frames support added

[*] Bug fixing (YUV-files, SSIM)

0.72

[+] AVS Support added

[*] Bug fixing

[!] Doesn't work with XviD

0.71

[+] First public beta

Main features

MSU Video Quality Measurement Tool PRO version provides many features for users. Here are some of them:

- **Two types of User Interface:**
  - Command-Line Interface;
  - Graphic User Interface;

- **Various input video formats:**
  - Video files (*.avi; *.avs; *.dat; *.divx; *.f4v; *.flv; *.h265; *.m2ts; *.m2v; *.m4v; *.mkv; *.mov; *.mp4; *.mpg; *.mts; *.mxf; *.ogm; *.ogv; *.qt; *.ts; *.vob; *.wmv; *.265; *.3g2; *.3gp; *.3gpp and others);
  - Raw files with 1-16 bit color depth (or 32-bit floats), different colorspaces and subsamplings;
  - Image files and image sequences: *.bmp, *.bw, *.cut, *.dds, *.exr, *.g3, *.gif (only static images), *.hdp, *.hdr, *.ico, *.iff, *.j2c, *.j2k, *.jif, *.jng, *.jp2, *.jpe, *.jpeg, *.jpg, *.jxr, *.lbm, *.mng, *.pbm, *.pcd, *.pct, *.pcx, *.pgm, *.pic, *.pict, *.png, *.ppm, *.psd, *.ras, *.rgb, *.rgba, *.sgi, *.targa, *.tga, *.tif, *.tiff, *.wap, *.wbm, *.wbmp, *.wdp, *.webp, *.xbm, *.xpm;
  - Full-HD, 4K, 8K and other resolutions support;
  - TIFF images with 16 and 32 bit (floating point) color depth;
  - AviSynth scripts – it can be very useful when using VOB, WMV and other files as input for MSU VQMT;
  - Video files via AviSynth scripts auto generation;

- **15 base metrics are included:**
  - PSNR;
  - PSNR (256);
  - APSNR;
  - APSNR (256);
  - MSE;
  - MSAD;
  - Delta;
  - SSIM (fast);
  - SSIM (precise);
  - MSSSIM (fast);
  - MSSSIM (precise);
  - 3SSIM;
  - ~~stSSIM~~ (temporary excluded);
  - VQM;
  - NIQE (no-reference);
  - VMAF;
  - ~~SI (no-reference)~~ (excluded till 12.1);
  - ~~TI (no-reference)~~ (excluded till 12.1);

- **7 additional objective quality metrics from MSU:**
  - MSU Brightness Flicking Metric (with source code);
  - MSU Brightness Independent PSNR (with source code);
  - MSU Drop Frame Metric (with source code);
  - MSU Noise Estimation Metric (with source code);
  - MSU Scene Change Detector (with source code);
  - MSU Blocking (no-reference);
  - MSU Blurring (no-reference);

- All metrics can be calculated simultaneously (in command-line mode);
- Every metric can be calculated for its set of color planes (Y, U, V, L, R, G, B, RGB, YUV);
- ROI;
- Results are saved to CSV of JSON files, visualization is available;
- Average values for each metric are calculated and saved;
- Plug-ins interface with SDK that gives user possibility to create their own metrics;
- Possibility to compare several files in one comparison – typical situation when comparing original uncompressed file with several compressed files with different codecs or presets;
- Possibility to save “bad” frames for every comparison with flexible options – it can be very useful when comparing two (or more) codecs and user wants to see frames where these codecs have maximum difference, the lowest/highest metric value, etc.

Command Line Options

The main advantage of MSU Video Quality Measurement Tool PRO version is the possibility to use command-line interface with flexible options for batch-processing.

The interface of MSU VQMT PRO version is

msu_metric.exe <parameters>

Comparison with older versions

- Specify metric to use and color component:

VQMT v.11, *one of:*	**Modern VQMT, *one of:***
-metr <metric name or metric alias> <color component>	-metr <metric name or metric alias>
	-metr <metric name or metric alias> over <color component>
	-metr <metric name or metric alias> over <color component1>, <color component2>,…, <color componentN>

Notes:

Since VQMT 12 color component can be skipped. VQMT will use the following algorithm:

- If metric supports only one color component, it will be selected.
- If metric supports none of (Y, U, V), it will be calculated over supported channels from (R, G, B).
- If metric supports none of (R, G, B), it will be calculated over supported channels from (Y, U, V).
- If all input files for metric (1 or 2) are in YUV color space, colors from set (Y, U, V) available for this metric will be used.
- If all input files for metric (1 or 2) are in RGB color space, colors from set (R, G, B) available for this metric will be used.
- If input files has different colorspaces, error is generated

Since VQMT 12 additional channels YUV and RGB added (supported by PSNR metric). To specify 3 separate channels, tell “Y,U,V”.

Get list of available metrics using “-list metrics”;

The list of available color spaces: Y, U, V, L, R, G, B, YUV, RGB. Use in metric list the list of color components, supported by each metric. Use can also use “-h <metric name>” for this.

- Running VQMT console:

VQMT v.9	**Modern VQMT, *one of:***
msu_metric_64 *(from VQMT install folder)*	vqmt *(from any folder in VQMT console)*
msu_metric_32 *(from VQMT install folder)*	
<*path to VQMT*>\msu_metric_64	
<*path to VQMT*>\msu_metric_32	

- Running VQMT GUI:

VQMT v.9	**Modern VQMT, *one of:***
N/A	-gui

- Specifying project file:

VQMT v.9	**Modern VQMT, *one of:***
N/A	-project <project file>

- Getting help:

VQMT v.4	Modern VQMT, *one of:*
<empty request>	<empty request>
	-h
	-?
	--help
	/h
	/?

- Specifying original file to use metric for:

VQMT v.4	Modern VQMT, *one of:*
-f <file name>	-f <file name> (*)
	-in <file name>
	-orig <file name>

(*) gray color means syntax is deprecated, can be removed in future releases

- Specifying files to use metric for:

VQMT v.4	Modern VQMT, *one of:*
-f <file name>	-f <file name>
	-in <file name>

Notes:

All input files should have the same resolution;

If files differ in length then the shortest length is taken for calculation: metrics are not calculated for the frames beyond the shortest file length;

You can specify JSON encoded open settings instead of <file name>, as it is in the GUI interface.

- Tell to use *all* available metrics with compatible color components:

VQMT v.4	Modern VQMT, *one of:*
-metr ALL	-metr ALL
	ALL

- Specify device to perform metric on:

VQMT v.4	Modern VQMT
N/A	-dev <device>

Notes:

Get list of available devices using “-list devices”.
<device> is device id or a part of device name.

- Specifying a mask file:

VQMT v.4	Modern VQMT
-mask <file name>	-mask <file name>

Such file is specified in the same way as described for **-in** parameter. It means that if a raw file is provided the user must specify data color space. Please read about available color spaces in the “-f” parameter description.

Notes:

A mask file should be a two color file: one color should mark masked area and another color should mark unmasked area. One of the colors should be black.

We assume the color with three zero components in RGB and with zero Y-component in YUV as “black”

- Tell to use black areas as masked (if mask file specified; default):

VQMT v.4	Modern VQMT
BLACK	BLACK

Note:

When black color is used as a mask an unmasked area may contain different colors – it is useful when the user wants to mark mask area on an actual video frame.

- Tell to use non-black areas as masked (if mask file specified):

VQMT v.4	Modern VQMT, *one of:*
NOT_BLACK	NOT_BLACK
	NOT BLACK

- Specify color space for raw file (*.yuv, etc.):

VQMT v.4	Modern VQMT
<color space>	<color space>

Notes:

Get list of available color spaces using “-list raw”

More information about *.yuv files can be found here: http://fourcc.org/yuv.php; PXXX color spaces are named according to Microsoft (http://msdn.microsoft.com/en-us/library/bb970578(VS.85).aspx) recommendations, but format is treated as in H264 standard (http://wiki.epfl.ch/amin/documents/itu-t%20h.264.pdf) streams definition.

Since VQMT 11 you can use standard names (like YUV420p), old VQMT names (like IYUV) and other aliases.

- Specify geometry transformation (resize or/and crop) for input files:

VQMT v.11.0	Modern VQMT, *one of:*
N/A	-resize [[<implementation>] <mode> [A]] to <dst> [crop]

Note: see next section for full documentation

- Specify resolution for raw file (*.yuv, etc.):

VQMT v.4	Modern VQMT, *one of:*
-yw <value>	-yw <value>
-yh <value>	-yh <value>
	<width>x<height>

Note: since VQMT v.11.1 resolution is individual for a last file. The first specified resolution will become default one for the files with not specified resolution.
Note: since VQMT v.5.0 resolution can be determined automatically from file name

- Specify open mode for reading file:

VQMT v.4	Modern VQMT
N/A	<open mode>

Notes:
Get list of available modes using “-list modes”.

The list of available modes:
- auto - Auto;
- avifile - AviFile;
- avisynth - AviSynth;
- avs_AviFileSource - AviSynth (AviFileSource);
- avs_AviSource - AviSynth (AviSource);
- avs_DirectShowSource - AviSynth (DirectShowSource);
- avs_OpenDMLSource - AviSynth (OpenDMLSource);
- ffmpeg - FFmpeg;
- raw - YUV raw file;
- sequence - Image sequence;
- y4m - Y4M file;

- Specify metric configuration line (for plug-in metrics):

VQMT v.4	Modern VQMT
-cl <config_line>	-cl <config_line>

Note:

Use this parameter if metric is configurable to set configuration string. Please refer to the metric homepage to find out applicable configuration line.

- Specify additional parameter for opening file:

VQMT v.4	Modern VQMT
N/A	-set <key> <value>

- Specify index creation mode:

VQMT v.4	Modern VQMT
N/A	-idx <index type>

Notes:

Index is needed if the source file cannot be correctly opened for seeking. Index file creation has a sense only if FFmpeg or Auto mode specified. When you use console, the file will be read frame by frame, and no index file will be created. Index file can be created when VQMT attempts to save bad frames.

Index type is one of:

- no - do not create index file (default);
- needed - create index file only if needed;
- forced - always create index file;
- needed_mem - build index in memory if needed, no files created;
- forced_mem - always build index in memory (no files created);

- Specify index file mask (default: *$(DirPath)\$(FileName).vqmtidx*):

VQMT v.4	Modern VQMT
N/A	-idx_file <file mask>

Note:

You can use placeholders *$(DirPath)* and *$(FileName)* in the mask.

- Turn on/off .csv file writing (default: *off*):

VQMT v.4, *one of:*	Modern VQMT, *one of:*
-sc 1	-sc 1
-sc 0	-sc 0
	-csv
	-csv yes
	-csv no

- Specify RGB to/from YUV table (default: *REC601*):

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-ryt REC601	-ryt REC601
-ryt PC601	-ryt PC601

- Specify .csv file name generation type (default: *POSTFIX*):

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-cng PREFIX	-cng PREFIX
-cng POSTFIX	-cng POSTFIX
-cng CUSTOM <filename>	-cng CUSTOM <file name>

- Specify directory for saving .csv files (default: current directory):

VQMT v.4	**Modern VQMT**
-cod <dir name>	-cod <dir name>

- Specify .log file directory:

VQMT v.4	**Modern VQMT, *one of:***
-lp <filename>	-lp <file name>
	-log <file name>

- Specify file for saving average results (old version):

VQMT v.4	**Modern VQMT**
-af <filename>	-af <file name>

- Specify file for saving average results:

VQMT v.4	**Modern VQMT, *one of:***
-af2 <filename>	-af2 <file name>
	-avg <file name>

- Specify .csv file delimiter (default: ,):

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-ct 0	-ct 0
-ct 1	-ct 1
	-ct ,
	-ct ;

- Specify floating point delimiter (default: .):

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-fpd 0	-fpd 0
-fpd 1	-fpd 1
	-fpd .
	-fpd ,

- Turn on/off saving visualization file (default: *off*):

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-sv 1	-sv 1
-sv 0	-sv 0
	-vis
	-vis yes
	-vis no

- Specify visualization method:

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
N/A	-vis-type <visualization type>

Note:
Try "-list visualizations" to see all available visualization methods and presets.

- Specify visualization method preset:

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
N/A	-vis-preset <visualization preset>

Note:
Try "-list visualizations" to see all available visualization methods and presets.

- Turn on/off saving bad frames:

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-sbf 1	-sbf 1
-sbf 0	-sbf 0
	-sbf
	-sbf yes
	-sbf no

- Specifying directory for bad frames:

VQMT v.4*:*	**Modern VQMT*:***
-bfod <directory>	-bfod <directory>

- Specifying type of bad frames:

VQMT v.4*:*	**Modern VQMT*:***
-bft <type>	-bft <type>

Note:

<type> is one of:

- ORIGPROC: save frames with max difference between first and second files;
- PROCPROC: save frames with max difference between second and third files (usually these files meant to be processed files to compare);
- FIRSTBETTER: save frames where second file is better than third;
- SECONDBETTER: save frames where third file is better than second;

- Specifying the number of bad frames:

VQMT v.4*:*	**Modern VQMT*:***
-bfnum <number>	-bfnum <number>

- Specifying the radius of bad frames (minimal distance between bad frames):

VQMT v.4*:*	**Modern VQMT*:***
-bfr <number>	-bfr <number>

Note:

0 means save bad frames by quality. 10 means that distance between any adjacent bad frames should be at least 10.

- Getting additional info:

VQMT v.4, *one of:*	**Modern VQMT, *one of:***
-list	-list
-list_raw	-list metrics
	-list modes
	-list colors
	-list raw
	-list devices
	-list visualizations

- Specifying the number of threads to use for processing:

VQMT v.4*:*	**Modern VQMT, *one of:***
N/A	-threads <number>

- Demanding JSON output:

VQMT v.4*:*	**Modern VQMT, *one of:***
N/A	-json

- Specifying file for writing JSON:

VQMT v.4*:*	**Modern VQMT, *one of:***
N/A	-json_file <file>

- Quiet mode:

VQMT v.4*:*	**Modern VQMT, *one of:***
N/A	-quiet

- Specifying frame range for input file:

VQMT v.4*:*	**Modern VQMT, *one of:***
N/A	-range <offset type> <range> -range <range> <range>

Note:

<offset type> is one of:

- seek: go to the first frame immediately, if possible;
- skip: read and skip some frames from beginning of input file (can be slow for big offsets);
- auto: *(default)* automatically choose between seeking and skipping.

<range> is one of:

- <number>-<number>: specifying the first and last frame of range;
- <number>-: set range from the first frame to the end of video.

The full list of available parameters

Help, information

command	description
-h **-?** **--help**	display help information about VQMT command line arguments;
-h <*metric*> **-?** <*metric*> **--help** <*metric*>	display extended information about specified metric;
-list **-list metrics** **-list modes** **-list colors**	display additional information;

command	description
-list raw **-list devices** **-list visualizations** **-list rgb-yuv**	

Input

command	description
-orig <*file*> **-in** <*file*>	specifying file to use metric for. Specify original file using **-orig**, distorted or other files using **-in**;
-stdin <*mode*> **-stdin-orig** <*mode*>	read raw (YUV) or Y4M data from standard input. Mode can be: **raw** or **y4m**. Use **-stdin-orig** for original file in case of reference metric;
<*color space*>	specify colorspace for RAW (.yuv) input file. View the list of available color spaces using **-list raw**;
<*number*>**x**<*number*>	specify resolution for last RAW (.yuv) input file. The first declaration becomes default for all other files Note: since VQMT v 5.0 resolution can be determined automatically from file name;
-range <*offset type*> <*range*> **-range** <*range*> <*range*>	specify frame range for last input file. <*offset type*> is one of : • **seek**: go to the first frame immediately, if possible; • **skip**: read and skip some frames from beginning of input file(can be slow for big offsets); • **auto**: (default) automatically choose between seeking and skipping. <*range*> is one of : • <*number*>-<*number*> : specifying the first and last frame of range; • <*number*>-: set range from the first frame to the end of

command	description
	video.
<open mode>	specify open mode for reading file. View the list of available color spaces using **-list modes**;
-no-upscale-uv **-no_upscale_uv yes** **-no_upscale_uv no**	measure subsampled planes if inputs are subsampled (Y422, Y422 formats for example). If off (default), inputs will be upscaled to resolution of Y-plane;

Metric

command	description
-metr *<metric name>* **[over** *<color components>***]** *<metric name>* **[over** *<color component>***]**	specifying metric and (optionally) color component; *<color component>* component or comma-separated list of components, for example **Y,U,V**, if *<color component>* is not specified, VQMT will use default component or components (based on metric supported components and source files colorspace); *<color components>* can be **ALL** - means all supported component for this metric. Use **-list metrics** to see all variants;
-cl *<config line>*	metric configuration line (applicable for plugin metrics only);
-set *<key>=<value>*	specify additional parameter for opening file or for metric configuration;
-dev *<device>*	device to perform metric on. *<device>* is device id or a part of device name. Get list of available devices using **-list devices**;

Output (JSON, CSV)

command	description
-json	perform JSON output;
-json-file *<file>*	write json to a file;
-json-default-file	write json to default file;
-json-fmt *<version>*	select JSON legacy format. Possible values: • 11

command	description
	• 12 (default);
-csv **-csv yes** **-csv no**	turn on .csv file writing. Use **-csv no** to turn off;
-cng PREFIX **-cng POSTFIX** **-cng CUSTOM** <*file name*>	.csv file name generation type;
-csv-dir <*dir name*>	dir for saving .csv files;
-log <*file name*>	log file;
-ct , **-ct ;**	.csv file delimiter (default: **,**);
-fpd . **-fpd ,**	.csv file floating point delimiter (default: **.**);

Visualization

command	description
-vis **-vis yes** **-vis no**	turn on/off saving visualization file;
-vng PREFIX **-vng POSTFIX**	visualization file name generation type;
-vis-type <*type*>	set visualization method. Use **-list visualizations** to see all methods;
-vis-preset <*preset*>	set visualization method preset. Use **-list visualizations** to see all presets;

command	description
-vis-caption-pos <*pos*>	set position of visualization caption where <*pos*> is **top** or **bottom**;
-vis-print-value yes **-vis-print-value no**	enable/disable printing value on visualization frames (default:yes);
-vis-print-frame yes **-vis-print-frame no**	enable/disable printing value on visualization frames (default:yes);
-vis-dir <*directory*>	set visualization output directory;

Mask

command	description
-mask <*file name*>	specifying a mask file. Such file is specified in the same way as described for **-in** parameter. It means that if a raw file is provided the user must specify data color space Notes: A mask file should be a two color file: one color should mark masked area and another color should mark unmasked area. One of the colors should be black. We assume the color with three zero components in RGB and with zero Y - component in YUV as **black**;
BLACK **NOT BLACK**	only when mask specified. BLACK means the program will use black areas as masked. Default: BLACK Note: when black color is used as a mask an unmasked area may contain different colors - it is useful when the user wants to mark mask area on an actual video frame;

Bad frames

command	description
-bad **-bad yes** **-bad no**	turn on/off saving bad frames;
-bad-dir <*directory*>	specify directory for bad frames;
-bad-type <*type*>	specify type of bad frames, where <*type*> is one of:

command	description
	• **ORIGPROC** : save frames with max difference between first and second files; • **PROCPROC** : save frames with max difference between second and third files(usually these files meant to be processed files to compare); • **FIRSTBETTER** : save frames where second file is better than third; • **SECONDBETTER** : save frames where third file is better than second;
-bad-format <*format*>	set bad frames image format. <*format*> is one of: **TIFF** (default) or **BMP**;
-bad-num <*number*>	specify the number of bad frames;
-bad-radius <*number*>	specify the radius of bad frames (minimal distance between bad frames);
-bad-name <*naming type*> **-bad-name CUSTOM** <*naming scheme*>	where <*naming type*> one of **frame**, **file**. Set mask for naming bad frames files: **frame** means - start file names from frame (default), **file** means - start file names from source file name (old-style). Or specify custom mask with placeholders;
-bad-thresold [<*lower threshold*>**]..[**<*upper threshold*>**]**	specify thresolds to skip bad frames for some metric values. Specify at least one threshold. Lower threshold means that metric must be better than this threshold, upper threshold means that metric must be worse than it (lower threshold can be bigger that upper one if lower values of metric means better quality);

Preprocessing

command	description
-resize [[<*implementation*>**]** <*mode*> **[A]] to** <*dst*> **[crop]**	correct geometry of inputs or downsample, where <*mode*> can be: • **crop** • **nearest**

command	description
	• **linear** • **cubic** (default) • **lanczos** and <*implementation*> is prefered algorithm implementation and can be: • **ffmpeg** (default) • **intel** and <*dst*> is destination size and can be: • **orig** - original size or 1-st input size • **1/2 orig** • **1/4 orig** • **min** - size of input with smallest resolution • **max** - size of input with biggest resolution • <*width*>**x**<*height*> **A** means antialising, can not be used with **crop** and **nearest**. If you specified **crop** after <*dst*>, the scaling will preserve ratio and than crop will be done. You can not specify **crop** after <*dst*> if <*mode*> is **crop**;

Indexing

command	description
-idx <*index type*>	specify index creation mode. One of: • **no** - do not create index file(default); • **needed** - create index file only if needed; • **forced** - always create index file; • **needed_mem** - build index in memory if needed, no files created; • **forced_mem** - always build index in memory(no files created); Default: **needed**;

command	description
	Note: Index is needed if the source file cannot be correctly opened for seeking. Index file creation has a sense only if FFmpeg or Auto mode specified. When you use console, the file will be read frame by frame, and no index file will be created. Index file can be created when VQMT attempts to save bad frames
-idx-file <*file mask*>	specify index file mask. Possible if index mode is one of: **needed**, **forced**. Default: **$(DirPath)\$(FileName).vqmtidx**;

RGB ↔ YUV, sample interpretation

command	description
-ryt REC601 **-ryt PC601**	rgb↔yuv table. Use **-list rgb-yuv** to see all variants;
-sample-conversion shift **-sample-conversion repeat**	integer samples interpretation. In the mode **shift** (default for VQMT 11 and earlier) 0xFF will be interpreted as 255/256;In the mode **repeat** (default since VQMT 12) 0xFF will be interpreted as 1;
-legacy-mode <*number*>	turn on legacy mode with specified VQMT version. NOTE: setting of other parameters of this group can broke reproducibility. Currently supported versions: 11;

Performance

command	description
-skip-frames fps <*fps*> **-skip-frames each** <*N*>	skip settings to increase performance. You can specify target <*fps*> and VQMT will try to acheive this fps, or you can process each <*N*>`th frame;
-slots <*number*>	number of image slots. 0 - default value. Increasing this value will sligt increase memory consumption, but can improve concurrency;
-metric-parallelism <*number*>	metric parallelism level - how many instances of single metric can be run simultaneously. 0 - default value. Set small value to reduce memory consumption. High metric parallelism is quite useless if you computing many metrics simultaneously;
-threads <*number*>	set number of threads to use for processing. 0 - default value;

command	description
-forced-length <*number*>	set length manually (0 to disable). Note: this parameter affects only progress bar, to limit frames count length use **-range 0-**<*last frame*> for a file;
-performance	output information about VQMT performance as JSON;

Misc.

command	description
-quiet	silent mode: do not output information about VQMT run;
-terminal	run VQMT in terminal mode;
-gui	run VQMT in GUI mode NOTE: this command allowed for GUI run;
-activate	activate VQMT license in terminal mode NOTE: this command allowed for GUI run;
-project <*file*>	load project, saved from GUI NOTE: this command allowed for GUI run;

Using JSON output

Introduction

VQMT PRO, running in console mode, can produce output in JSON format to standard output or file, which contains:

- Information about VQMT program, including version.
- Information about opened files and used metrics.
- Results of metrics calculation.
- Average results of metrics.

JSON file will be formed partially as corresponding information are calculated.

Scheme of JSON file

JSON output is a JSON-object, which contains fields “head”, “values” and “avg”.

The field “head” consists of fields “generator” (information about VQMT), “files” (information about opened files) and “metrics” (information about used metrics).

The field “values” is array. Each element of this array contains metrics values, corresponding to one frame. Element is array itself with length, equal to length of array “metrics” from “head” field of root object.

Output is produced with corresponding to the following scheme:

```
{
 "head" : {
  "generator" : {
     "program"           : "<full program name and
version>",
     "program_shortname" : "VQMT",
     "company"           : "MSU Graphics & Media Lab Video
Group",
     "project_leader"    : "Dr. Dmitriy Vatolin",
     "build_date"        : "<build date>",
     "url"               : "http://compression.ru/video/",
     "edition"           : "<PRO/DEMO>",
     "ver_maj"           : <version major>,
     "ver_min"           : <version minor>,
     "ver_special"       : "<version special>",
     "ver_rev"           : <revision>
  },
```

```
  "files" : [
     {
          "index"      : <number>,
          "path"       : "<1st file path>",
          "error"      : <true/false>,
          "colorspace" : "<colorspace>",
          "codec"      : "<codec>",
          "size"       : [<width>,<height>],
          "length"     : {"frames" : <expected frames in
files>,"isExact" : <true/false> },
          "open_mode"  : "<open mode>",
          "original"   : <true/false>,
          "fps"        : <null or floating point>

     }, ...
  ],

  "metrics" : [
     {
          "metric_name"      : "<1st metric name>",
          "metric_variation" : "<metric variation>",
          "metric_aliases"   : ["<metric alias>",...],
          "info_url"         : "<info_url>",
          "device"           : "<device_name>",
          "color_component"  : "<color component>",
          "compaired_files"  : [<file index>, ...],
          "config_summary"   : "",
          "uv-scale"         : {"expected" : {"x" :
     <number>,"y" : <number>},"real" : {"x" : <number>,"y" :
     <number>}},
          "value_id"         : "<text identifier of result
to distinguish different results of single metric>",
          "col"              : "<column name>"

     }, ...
  ]
 },

 "values" : [
   {"frame": <frame number>, "data": {"<column 1 name>":
<null or value of metric>, ...}}, ...
 ],

 "accumulated" : {
   "accumulator 1 name": {"<column 1 name>": <null or
accumulated value>, ... }, ...
 }
}
```

Example of JSON output

Below there is a probable content of file test.json after executing the following command:

```
vqmt -in c:\video\cinepack.avi -in c:\video\loseless.avi 0-
10 -metr PSNR -metr SSIM over Y -json_file test.json
```

Content of the file:


```
{
  "generator": {
    "program"          : "MSU Video Quality Measurement Tool PRO 12.0
r12383 beta",
    "program_shortname" : "VQMT",
    "company"          : "MSU Graphics & Media Lab Video Group",
    "build_date"       : "Oct 31 2019",
    "url"              : "http://compression.ru/video/",
    "project_leader"   : "Dr. Dmitriy Vatolin",
    "edition"          : "PRO",
    "json_fmt"         : 12,
    "ver_maj"          : 12,
    "ver_min"          : 0,
    "ver_special"      : "beta",
    "ver_rev"          : 12383,
    "comment"          : "use '-json-fmt 11' in command line for legacy
json file"
  },
  "head": {
    "files": [
      {
        "index"      : 0,
        "path"       : "c:\\video\\loseless.avi",
        "error"      : false,
        "colorspace" : "RGB24",
        "codec"      : "rgb24 -> RGB24 format AVI (Audio Video
Interleaved) codec Lagarith lossless",
        "size"       : [1280,720],
        "length"     : {"frames" : 51,"isExact" : false,"isAeterna" :
false},
        "original"   : true,
        "open_mode"  : "auto (FFmpeg)",
        "fps"        : 22.270729270729269,
        "idx"        : 0
      },
      {
        "index"      : 1,
        "path"       : "c:\\video\\cinepack.avi",
        "error"      : false,
        "colorspace" : "RGB24",
        "codec"      : "rgb24 -> RGB24 format AVI (Audio Video
Interleaved) codec Cinepak",
        "size"       : [1280,720],
        "length"     : {"frames" : 10,"isExact" : false,"isAeterna" :
false},
        "original"   : false,
        "open_mode"  : "auto (FFmpeg)",
        "fps"        : 22.270729270729269,
        "idx"        : 1
      }
```

```
    ],
    "metrics": [
      {
        "metric_name"      : "psnr",
        "metric_variation" : "",
        "metric_aliases"   : ["apsnr"],
        "info_url"         :
"http://www.compression.ru/video/quality_measure/info.html#psnr",
        "device"           : "CPU",
        "color_component"  : "R",
        "compaired_files"  : [0,1],
        "config_summary"   : "",
        "uv-scale"         : {"expected" : {"x" : 1,"y" : 1},"real" :
{"x" : 1,"y" : 1}},
        "value_id"         : "",
        "col"              : "A"
      },
      {
        "metric_name"      : "psnr",
        "metric_variation" : "",
        "metric_aliases"   : ["apsnr"],
        "info_url"         :
"http://www.compression.ru/video/quality_measure/info.html#psnr",
        "device"           : "CPU",
        "color_component"  : "G",
        "compaired_files"  : [0,1],
        "config_summary"   : "",
        "uv-scale"         : {"expected" : {"x" : 1,"y" : 1},"real" :
{"x" : 1,"y" : 1}},
        "value_id"         : "",
        "col"              : "B"
      },
      {
        "metric_name"      : "psnr",
        "metric_variation" : "",
        "metric_aliases"   : ["apsnr"],
        "info_url"         :
"http://www.compression.ru/video/quality_measure/info.html#psnr",
        "device"           : "CPU",
        "color_component"  : "B",
        "compaired_files"  : [0,1],
        "config_summary"   : "",
        "uv-scale"         : {"expected" : {"x" : 1,"y" : 1},"real" :
{"x" : 1,"y" : 1}},
        "value_id"         : "",
        "col"              : "C"
      },
      {
        "metric_name"      : "ssim",
        "metric_variation" : "fast",
        "metric_aliases"   : ["ssim_fast"],
        "info_url"         :
"http://www.compression.ru/video/quality_measure/info.html#ssim",
        "device"           : "CPU",
        "color_component"  : "Y",
        "compaired_files"  : [0,1],
        "config_summary"   : "",
        "uv-scale"         : {"expected" : {"x" : 1,"y" : 1},"real" :
{"x" : 1,"y" : 1}},
```

```
        "value_id"         : "",
        "col"              : "D"
      }
    ]
  },
  "values": [
    {"frame" : 0,"data" : {"A" : 34.941802978515625,"B" :
37.199314117431641,"C" : 35.543128967285156,"D" :
0.94766181707382202,"bps0" : 69836.375000000000,"bps1" :
36159.468750000000}},
    {"frame" : 1,"data" : {"A" : 37.363426208496094,"B" :
39.718318939208984,"C" : 37.873035430908203,"D" :
0.97358179092407227,"bps0" : 49778.820312500000,"bps1" :
27564.748046875000}},
    {"frame" : 2,"data" : {"A" : 35.978748321533203,"B" :
38.394237518310547,"C" : 36.248638153076172,"D" :
0.96748113632202148,"bps0" : 52109.410156250000,"bps1" :
25277.455078125000}},
    {"frame" : 3,"data" : {"A" : 36.695510864257813,"B" :
39.391960144042969,"C" : 36.536998748779297,"D" :
0.97264289855957031,"bps0" : 55631.210937500000,"bps1" :
25632.361328125000}},
    {"frame" : 4,"data" : {"A" : 36.842735290527344,"B" :
38.951107025146484,"C" : 35.710212707519531,"D" :
0.96764451265335083,"bps0" : 57655.000000000000,"bps1" :
27458.205078125000}},
    {"frame" : 5,"data" : {"A" : 36.373535156250000,"B" :
39.171756744384766,"C" : 36.922657012939453,"D" :
0.97091424465179443,"bps0" : 60703.058593750000,"bps1" :
27695.523437500000}},
    {"frame" : 6,"data" : {"A" : 36.462524414062500,"B" :
39.036079406738281,"C" : 37.319820404052734,"D" :
0.97270447015762329,"bps0" : 49801.804687500000,"bps1" :
30017.736328125000}},
    {"frame" : 7,"data" : {"A" : 35.649997711181641,"B" :
38.739669799804688,"C" : 37.214542388916016,"D" :
0.97023040056228638,"bps0" : 58100.058593750000,"bps1" :
29694.542968750000}},
    {"frame" : 8,"data" : {"A" : 36.551681518554688,"B" :
39.322135925292969,"C" : 36.609455108642578,"D" :
0.97235065698623657,"bps0" : 52321.605468750000,"bps1" :
33312.378906250000}},
    {"frame" : 9,"data" : {"A" : 36.585155487060547,"B" :
39.076419830322266,"C" : 36.979415893554688,"D" :
0.97408014535903931,"bps0" : 57775.261718750000,"bps1" :
31875.292968750000}}
  ],
  "accumulated": {
    "total psnr"    : {"A" : 36.295528411865234,"B" :
38.845626831054688,"C" : 36.641197204589844},
    "mean"          : {"A" : 36.344512939453125,"B" :
38.900100708007813,"C" : 36.695789337158203,"D" : 0.96892923116683960},
    "harmonic mean" : {"A" : 36.333053588867188,"B" :
38.888519287109375,"C" : 36.682914733886719,"D" : 0.96887147426605225},
    "min. val"      : {"A" : 34.941802978515625,"B" :
37.199314117431641,"C" : 35.543128967285156,"D" : 0.94766181707382202},
    "max. val"      : {"A" : 37.363426208496094,"B" :
39.718318939208984,"C" : 37.873035430908203,"D" : 0.97408014535903931},
    "min. frame"    : {"A" : 0,"B" : 0,"C" : 0,"D" : 0},
```

```
    "max. frame"    : {"A" : 1,"B" : 1,"C" : 1,"D" : 9},
    "std dev"       : {"A" : 0.64100986719131470,"B" :
0.66255629062652588,"C" : 0.68621718883514404,"D" :
0.0074135810136795044},
    "variance"      : {"A" : 0.41089364886283875,"B" :
0.43898084759712219,"C" : 0.47089403867721558,"D" : 5.4961183195700869e-
05}
  }
}
```

Information about metrics

Basic information

In VQMT since version 12.0 we consider the range of all inputs to be 0..1. Each channel will be brought to this range in accordance with the settings.

VQMT has setting *Sample conversion* which determines how the conversion of integer to 0..1 range will be performed. This setting has 2 values:

- *shift* we consider 2^n, where n is sample bitness, is converted to 1. Note, that the maximum 1 is unreachable. In this mode real reachable maximum depends from the bitness of source image. For example, for 8-bit input it will be 255/256, for 16-bit input it will be 65535/65536. This method corresponds to some standards of HDR TV's implementation, and this is default for some metrics of earlier versions of VQMT. This method is bad for inputs with low bitness. For example, maximum of 1-bit monochrome image will be interpreted as 0.5, which is gray color, not white.
- *repeat tail* we consider $2^n - 1$, where n is sample bitness, is converted to 1. In this mode real reachable maximum is always 1, the range not depends from input bitness.

The real range of input could be also shrunk if RGB ↔ YUV conversion performed. PC conversion tables do not modify the range, but REC do.

Overview

Current version of MSU VQMT PRO contains 16 implemented objective quality metrics:

- **Universal reference objective quality metrics** (for estimation the similarity between two or several images/video):
  - Delta
  - MSE
  - MSAD
  - PSNR
  - SSIM Index
  - MultiScale SSIM Index
  - 3SSIM Index
  - Spatio-Temporal SSIM Index (Temporary excluded)
  - VQM
  - VMAF
- **Universal no-reference objective quality metrics** (for estimation the naturality of images/video without reference):
  - NIQE
- **Specific non-reference objective metrics for special types of artifacts** (for estimation the desired artifacts)

  - MSU Blurring
  - MSU Blocking
  - MSU Brightness Flicking Metric
  - MSU Drop Frame Metric
  - MSU Noise Estimation Metric
  - SI (Temporary excluded)
  - TI (Temporary excluded)
- **Special reference PSNR metric edition** to compare videos regardless on the different brightness
  - MSU Brightness Independent PSNR

Delta

Brief Description

The value of this metric is the mean difference of the color components in the correspondent points of image. This metric is used for checking codecs/filters for errors like losses or growths of luminance, not for quality comparisons.

$$d(X,Y) = \frac{\sum_{i=1,j=1}^{m,n}(X_{i,j} - Y_{i,j})}{mn}$$

The values are in -1..1. 0 – for identical frames (value 0 can be a metric value is frames have considerable differences).

Note:

In legacy mode 11, value is in range -255..255 for channels R, G, B, Y, U, V, -100..100 for channel L.

Examples

Here is example of this metric

First frame Second frame Delta

Picture 1. Delta example for two frames

Here are more examples how different distortions have influence on Delta value.

Original image

Image with added noise

Blurred image

Sharpen image

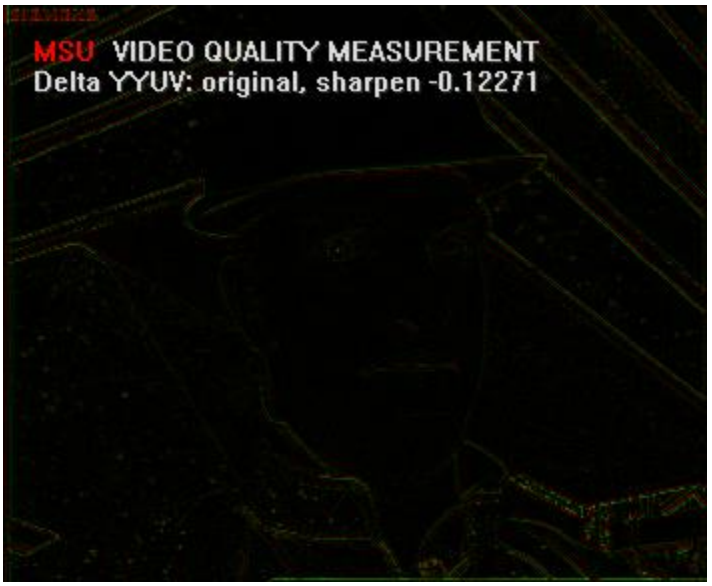
Picture 2. Original and processed images (for Delta example)

And here are the delta values of Y–plane for these images

Delta for image with itself, value = 0

Delta for image with noisy image, value = 0.0971987

Delta for image with blurred image, value = 0.0296287



Delta for image with sharpen image, value = -0.12271

Picture 3. Delta values for original and processed images (for Delta example)

MSE

Brief description

MSE is one metric used to assess how well a method to reconstruct an image performs relative to the original image. It shows mean square error for two images or frames.

$$d(X,Y) = \frac{\sum_{i=1,j=1}^{m,n}(X_{i,j} - Y_{i,j})^2}{mn}$$

The values are in range 0..1. 0 – for identical frames.

Note:

In legacy mode 11, value is in range 0.. 65536 for channels R, G, B, Y, U, V, 0..10000 for channel L.

Examples

Here is example of this metric:

Original Processed MSE

Picture 4. MSE example for two frames

Here are more examples how different distortions influence on MSE value.

Original image

Image with added noise

Blurred image

Sharpen image

Picture 5. Original and processed images (for MSE example)

And here are the MSE values of Y–plane for these images

MSE for image with itself, value = 0

MSE for image with noisy image, value = 161.968

MSE for image with blurred image, value = 55.2885

MSE for image with sharpen image, value = 0.958917

Picture 6. MSE values for original and processed images (for MSE example)

MSAD

The value of this metric is the mean absolute difference of the color components in the correspondent points of image.

$$d(X,Y) = \frac{\sum_{i=1,j=1}^{m,n} \left| X_{i,j} - Y_{i,j} \right|}{mn}$$

The values are in 0..1. 0 – for identical frames.

Note:

In legacy mode 11, value is in range 0..256 for channels R, G, B, Y, U, V, 0..100 for channel L.

Examples

Here is example of this metric visualization:

Original Processed MSAD

Picture 7. MSAD example for two frames

Here are more examples how different distortions have influence on MSAD value.

Original image

Image with added noise

Blurred image

Sharpen image

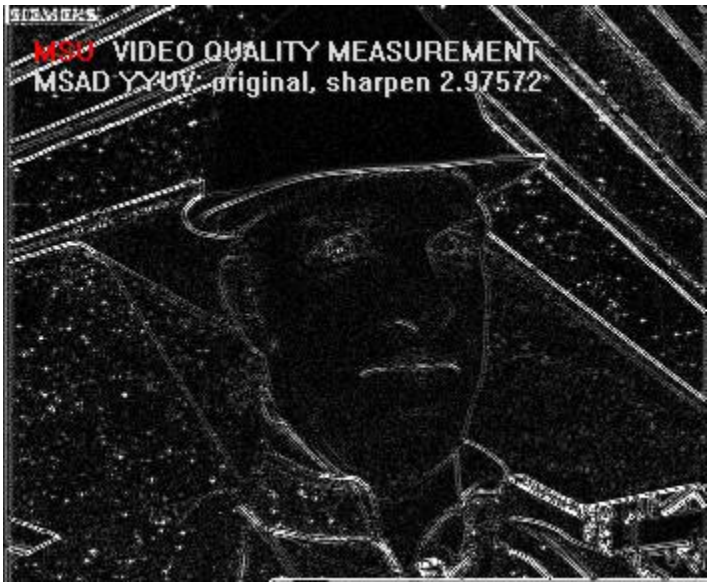
Picture 8. Original and processed images (for MSAD example)

And here are the MSAD values of Y–plane for these images

MSAD for image with itself, value = 0

MSAD for image with noisy image, value = 10.2417

MSAD for image with blurred image, value = 3.77253



MSAD for image with sharpen image, value = 2.97572

Picture 9. MSAD values for original and processed images (for MSAD example)

PSNR (Peak Signal-to-Noise Ratio)

Brief Description

This metric, which is used often in practice, called peak-to-peak signal-to-noise ratio — PSNR.

$$PSNR = 10 \cdot \log_{10} \frac{MaxErr^2 \cdot w \cdot h}{\sum_{i=0,o=0}^{w,h} (x_{ij} - y_{ij})^2},$$

where *MaxErr* – maximum possible absolute value of color components difference, *w* – video width, *h* – video height. Generally, this metric is equivalent to Mean Square Error, but it is more convenient to use because of logarithmic scale. It has the same disadvantages as the MSE metric.

In MSU VQMT you can calculate PSNR for all YUV and RGB components and for L component of LUV color space. Also, since VQMT 12 you can calculate over all YUV space or all RGB space, achieving a single value for 3 components. PSNR metric is easy and fast to calculate, but sometimes it is not appropriate to human's perception.

In VQMT 12 all input samples are implicitly cast to interval 0..1, so MaxErr is always 1. In VQMT 11 and earlier PSNR metric could consider real maximum value that could appear after all sample conversions and transformations for MaxErr. VQMT 12 will be use same behavior in Legacy mode. To simulate this, you also could make the following steps (without Legacy mode):

- Set sample conversion mode to "Repeat",
- Set RGB ↔ YUV table to PC range.

Modern PSNR metric is quite similar to PSNR256 metric of earlier VQMT.

Also, in the old version, there were metrics APSNR and APSNR256, which differed from the corresponding metrics by the method of calculating the average. In A- metrics it was arithmetic mean. In common PSNR, PSNR256 it was total PSNR (computing total MSE for all processed frames). In VQMT 12 both average values will be calculated simultaneously, so the need for A-metrics has disappeared.

Examples

Here is example of this metric:

Original Processed PSNR

Picture 10. PSNR example for two frames

Here are more examples how different distortions have influence on PSNR value.

Original image

Image with added noise

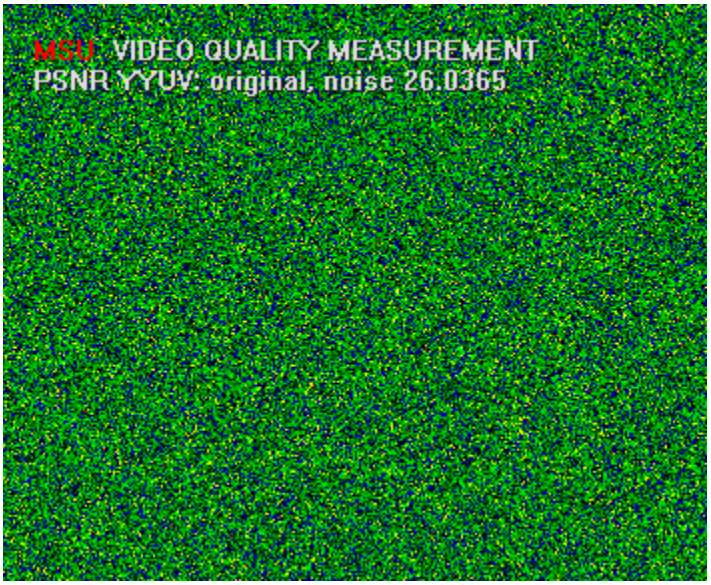
Blurred image

Sharpen image

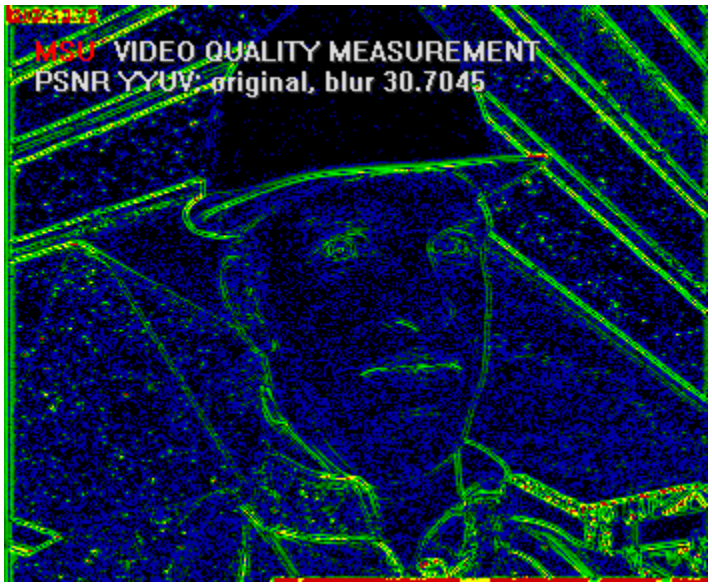
Picture 11. Original and processed images (for PSNR example)

And here are the PSNR values of Y–plane for these images

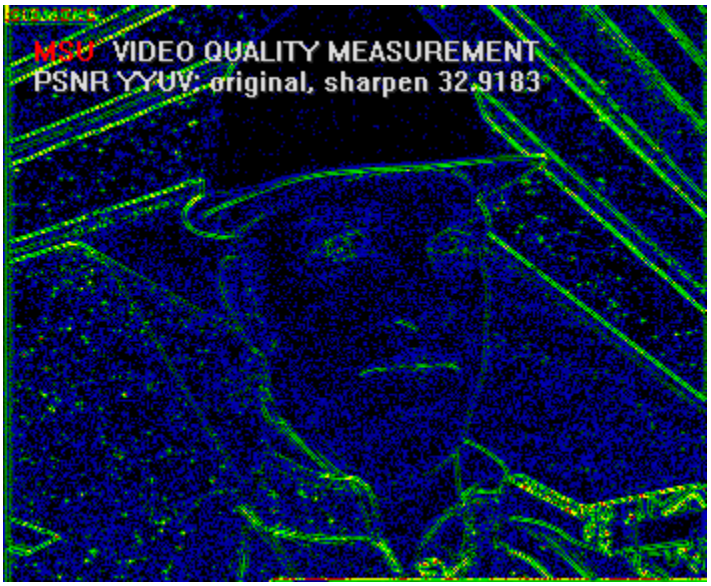
PSNR for image with itself, value = 0



PSNR for image with noisy image, value = 26.0365



PSNR for image with blurred image, value = 30.7045



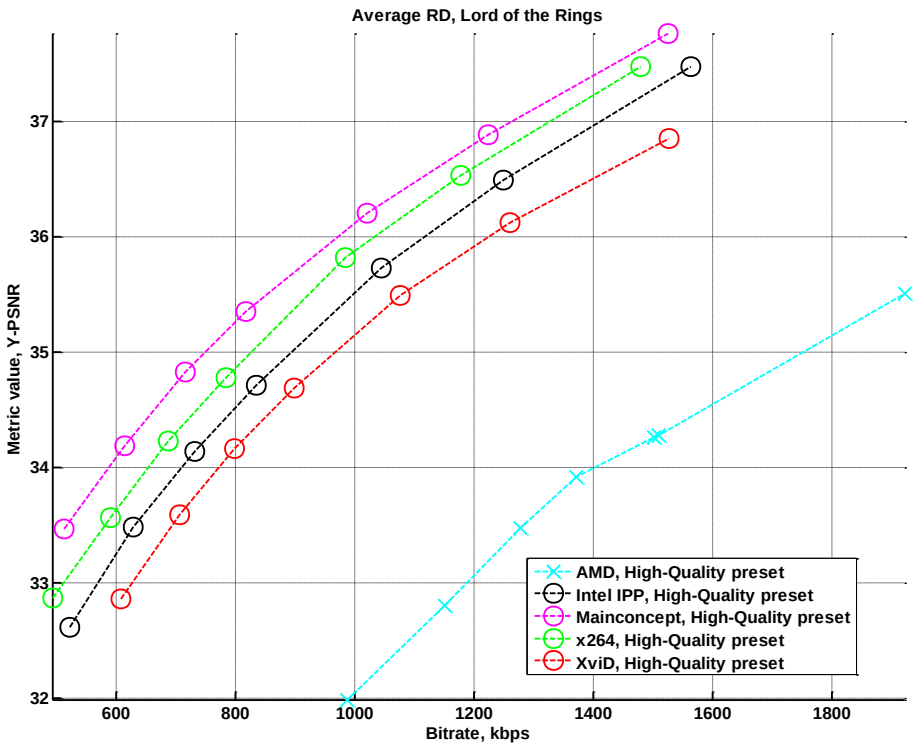
PSNR for image with sharpen image, value = 32.9183

Picture 12. PSNR values for original and processed images (for PSNR example)

Also MSU VQMT with PSNR metric was widely used during all MSU Codec Comparisons, including:

- MSU MPEG-4 SP/ASP Codec Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-4_en.html
- MSU JPEG 2000 Image Codecs Comparison
  http://www.compression.ru/video/codec_comparison/jpeg2000_codecs_comparison_en.html
- First Annual MSU H.264/MPEG-4 AVC Video Codec Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_en.html
- Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2005_en.html
- MPEG-2 Video Decoders Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-2_2006_en.html
- MSU Windows Media Photo (Microsoft HD Photo) and JPEG 2000 Codec Comparison
  http://www.compression.ru/video/codec_comparison/wmp_codecs_comparison_en.html
- Third Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2006_en.html
- Lossless Video Codec Comparison '2007
  http://www.compression.ru/video/codec_comparison/lossless_codecs_2007_en.html
- Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
  http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2007_en.html

Here is an example of RD-curve from Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison:



Picture 13. PSNR Example for RD-curve

SSIM INDEX

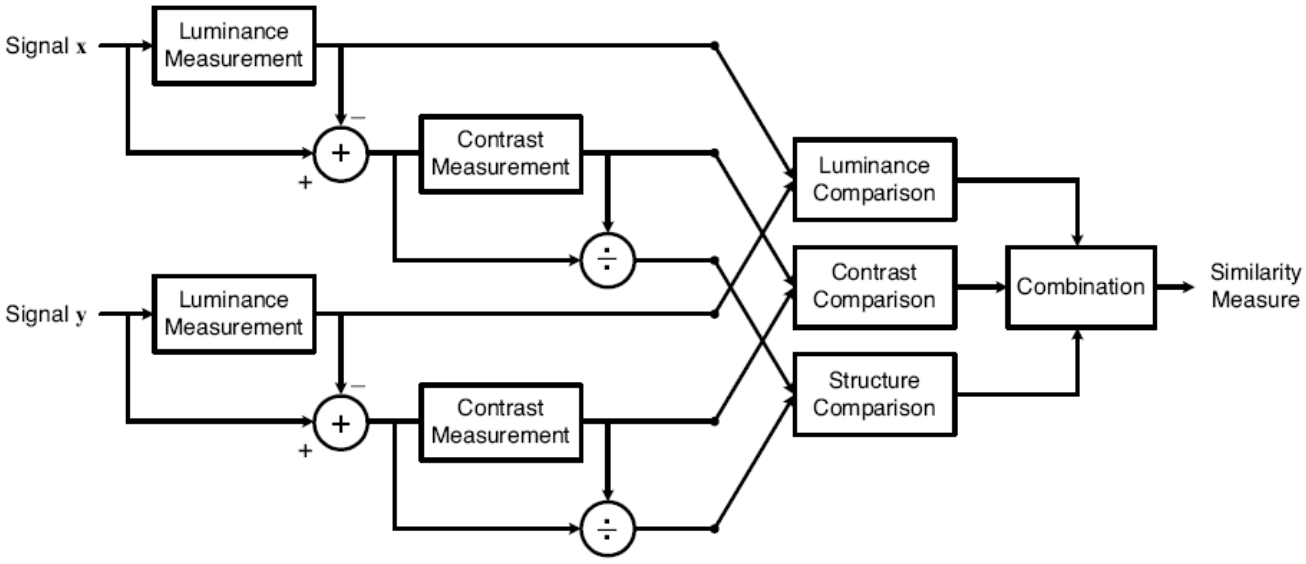
Brief Description

Original paper is Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli, *"Image Quality Assessment: From Error Visibility to Structural Similarity"*, IEEE Transactions on Image Processing, Vol. 13, No 4, April 2004.

This article could be found here:
http://ieeexplore.ieee.org/iel5/83/28667/01284395.pdf

SSIM author's homepage: http://www.cns.nyu.edu/~lcv/ssim/

The scheme of SSIM calculation could be presented as:



Picture 14. Diagram of the structural similarity (SSIM) measurement system[1]

Main idea of the structure similarity index (SSIM) is to compare distortion of three image components:

- Luminance comparison
- Contrast comparison
- Structure comparison

Final formula after combination of these comparisons is the following:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x + \mu_y + C_1)(\sigma_x + \sigma_y + C_2)}$$

where

[1] This scheme is taken from paper Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", IEEE Transactions on Image Processing, Vol. 13, No 4, April 2004.

$$\mu_x = \sum_{i=1}^{N} \omega_i x_i$$

$$\sigma_x = \left(\sum_{i=1}^{N} \omega_i (x_i - \mu_x) \right)^{\frac{1}{2}}$$

$$\sigma_{xy} = \sum_{i=1}^{N} \omega_i (x_i - \mu_x)(y_i - \mu_y)$$

In our program constants C_1 and C_2 are calculated using the following expressions:

- *C1 = 0.01 * 0.01*
- *C2 = 0.03 * 0.03*

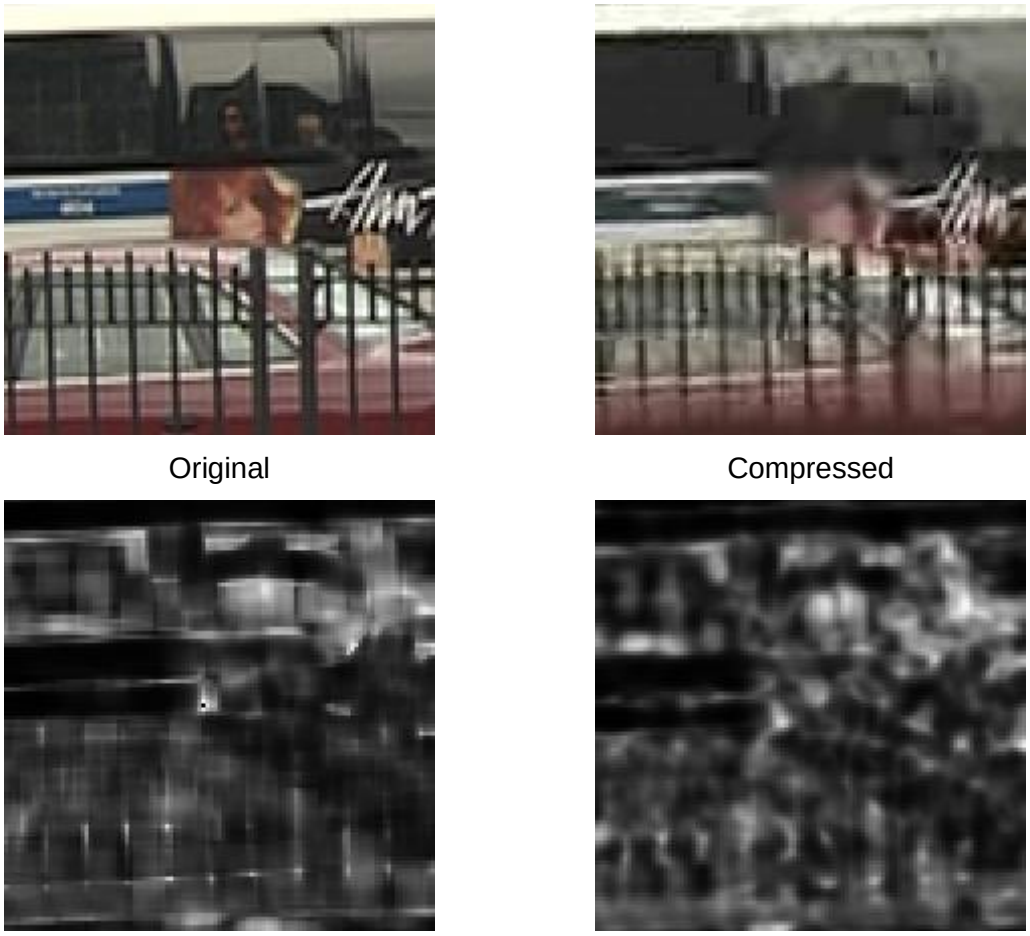
There are 2 implementation of SSIM in our program: fast and precise. The fast one is equal to our previous SSIM implementation. The difference is that the fast one uses box filter, while the precise one uses Gaussian blur.

Fast implementation visualization seems to be shifted. This effect is causes by the sum calculation algorithm for the box filter. The sum is calculated over the block to the bottom-left or up-left of the pixel (depending on if the image is bottom-up or top-down).

In our implementations one SSIM value corresponds to two sequences. Values are in range -1…1. Higher values are better, 1 for equal frames. The advantages of SSIM metric is that it is more closer to human's vision, than PSNR, but it is more complex and takes more time to calculate.

Examples

Here is an example of SSIM result for original and processed (compressed with lossy compression) images.



Original Compressed

SSIM (fast) visualization SSIM (precise) visualization

Picture 15. SSIM example for compressed video

Here are more examples how different distortions have influence on SSIM value.

Original image

Image with added noise

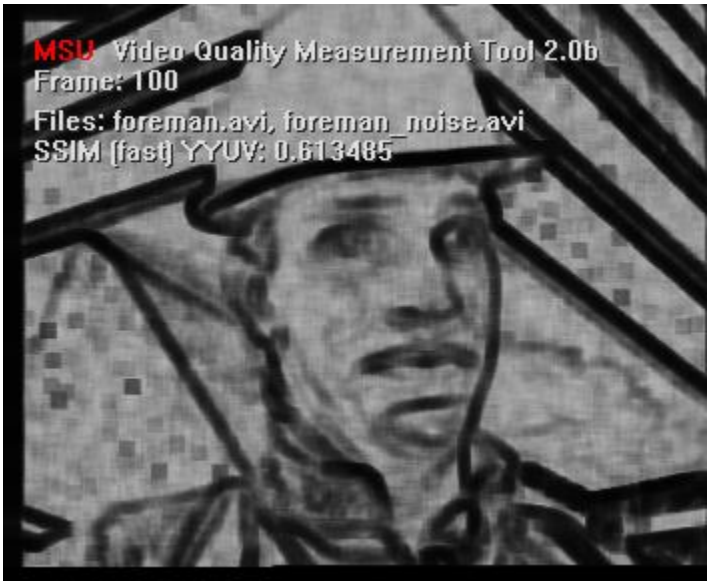
Blurred image

Sharpen image

Picture 16. Original and processed video sequences (for SSIM example)

Here are the Y-plane SSIM (fast) visualizations of these video sequences.

SSIM for image with itself, value = 1



SSIM (fast) for image with noisy image, value = 0.61348

SSIM (fast) for image with blurred image, value = 0.93457

SSIM (fast) for image with sharpen image, value = 0.97832

Picture 17. SSIM (fast) visualizations for original and processed video sequences

Here are the Y-plane SSIM (precise) visualizations of these video sequences.

SSIM for image with itself, value = 1

SSIM (precise) for image with noisy image, value = 0. 581786

SSIM (precise) for image with blurred image, value = 0.923186

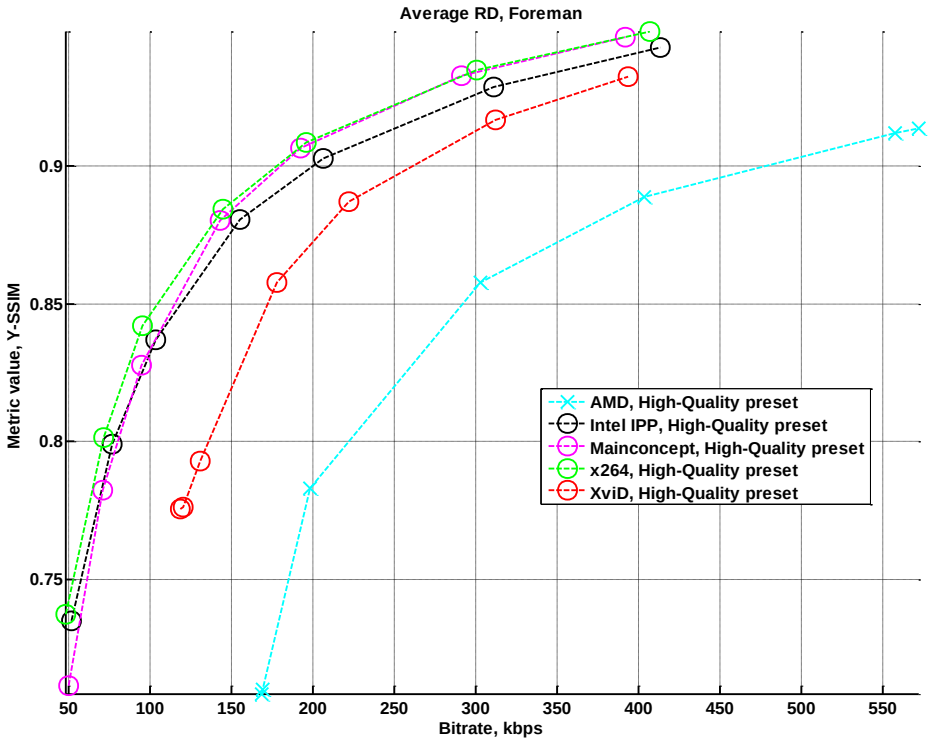
SSIM (precise) for image with sharpen image, value = 0.976667

Picture 18. SSIM (precise) visualizations for original and processed video sequences

Also MSU VQMT with SSIM metric was widely used during MSU Codec Comparisons, including:

- Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2005_en.html
- Third Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2006_en.html
- Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2007_en.html

Here is an example of RD-curve from Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison:



Picture 19. SSIM Example for RD-curve

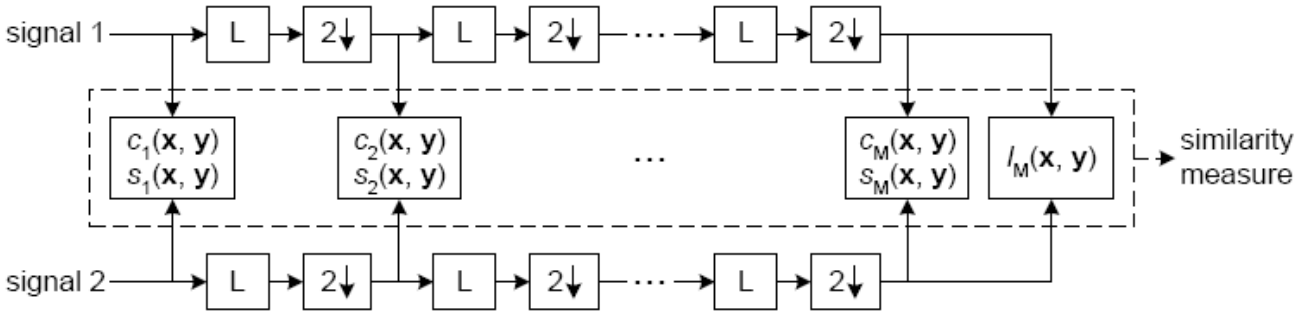
MultiScale SSIM INDEX

Brief Description

Original paper is *Zhou* Wang, Eero P. Simoncelli and Alan C. Bovik, *"MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT"*, 37th IEEE Asilomar Conference on Signals, Systems and Computers, Nov. 2003.

This article could be found here:
http://ieeexplore.ieee.org/iel5/9072/28784/01292216.pdf

The scheme of MSSSIM calculation could be presented as:



Picture 20. Diagram of the multiscale structural similarity (MSSSIM) measurement system[2]

Main idea of the multiscale structure similarity index (SSIM) is to compare distortion of three image components:

- Luminance comparison
- Contrast comparison
- Structure comparison

as it was done in SSIM index. But now we calculate similarity for several levels – each is downsampled version of previous level. First level is original image. Overall metric value over M levels is calculated as:

$$MSSSIM(x,y) = [l_M(x,y)]^{\alpha_M} * \prod_{j=1}^{M} [c_j(x,y)]^{\beta_j} [s_j(x,y)]^{\gamma_j} ,$$

where

$l_i(x,y) = \dfrac{(2\mu_x\mu_y + C_1)}{(\mu_x + \mu_y + C_1)}$ – luminance measurement for level i

[2] This scheme is taken from paper Zhou Wang, Eero P. Simoncelli and Alan C. Bovik, "MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT", 37th IEEE Asilomar Conference on Signals, Systems and Computers, Nov. 2003.

$$c_i(x,y) = \frac{(2\sigma_x\sigma_y + C_2)}{(\sigma_x^2 + \sigma_y^2 + C_2)}$$ – contrast measurement for level i

$$s_i(x,y) = \frac{(\sigma_{xy} + C_3)}{(\sigma_x\sigma_y + C_3)}$$ – structure measurement for level i

$$\mu_x = \sum_{i=1}^{N} \omega_i x_i$$

$$\sigma_x = \left(\sum_{i=1}^{N} \omega_i (x_i - \mu_x) \right)^{\frac{1}{2}}$$

$$\sigma_{xy} = \sum_{i=1}^{N} \omega_i (x_i - \mu_x)(y_i - \mu_y)$$

and $\alpha_i, \beta_i, \gamma_i$ – weights for appropriate fractions in i-level of our measurement.

In our program constants C_1, C_2 and C_3 are calculated using the following expressions:

- *C1 = 0.01 * 0.01*
- *C2 = 0.03 * 0.03*
- *C3 = 0.5*C2*

where *video1Max* is the maximum value of a given color component for the first video, *video2Max* is the maximum value of the same color component for the second video. Maximum value of a color component is calculated in the same way as for

PSNR (Peak Signal-to-Noise Ratio):

- *videoMax* = 255 for 8 bit color components
- *videoMax* = 255 + 3/4 for 10 bit color components
- *videoMax* = 255 + 63/64 for 14 bit color components
- *videoMax* = 255 + 255/256 for 16 bit color components

Same as in SSIM, there are two implementation of MSSSIM in our program: fast and precise. The fast one is implemented in the same way as it was done for SSIM metric. The difference between fast and precise is that the fast one uses box filter, while the precise one uses Gaussian blur.

Fast implementation visualization seems to be shifted. This effect is caused by the sum calculation algorithm for the box filter. The sum is calculated over the block to the bottom-left or up-left of the pixel (depending on the image orientation: bottom-up or top-down). Also result pixels in visualization is quite dark due to it's value is result of all levels measure values multiplication, that are between 0.0 and 1.0.

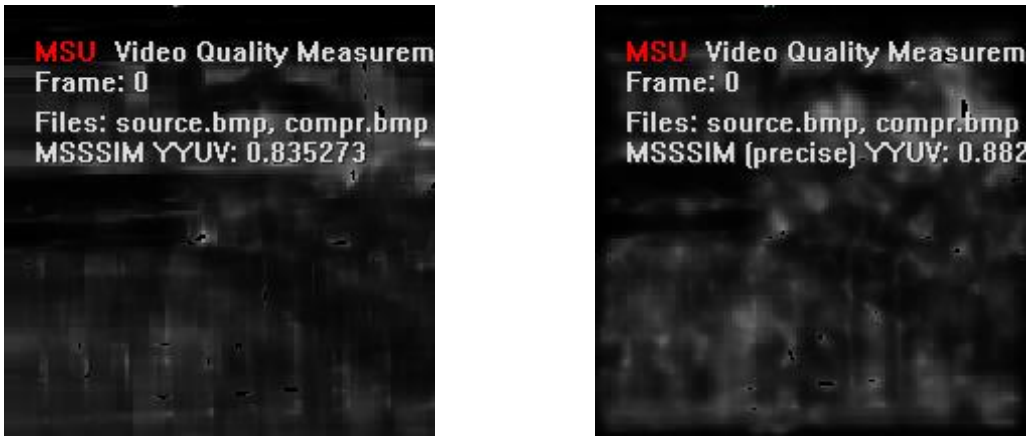
In our implementations one MSSSIM value corresponds to two sequences. Values are in range -1…1. The higher values the better video quality, 1 for equal frames. The advantages of SSIM metric is that it is more closer to human's vision system, than PSNR, but it is more complex and takes more time to calculate.

Examples

Here is an example of MSSSIM result for original and processed (lossy compression) images.

Original

Compressed



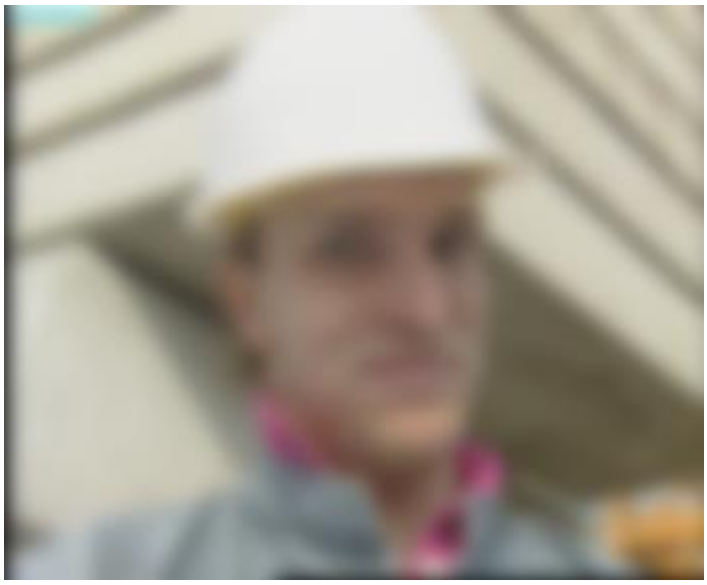
MSSSIM (fast) visualization MSSSIM (precise) visualization

Picture 21. MSSSIM example for compressed video

Here are more examples how different distortions have influence on MSSSIM value.

Original image

Image with added noise



Blurred image

Sharpen image

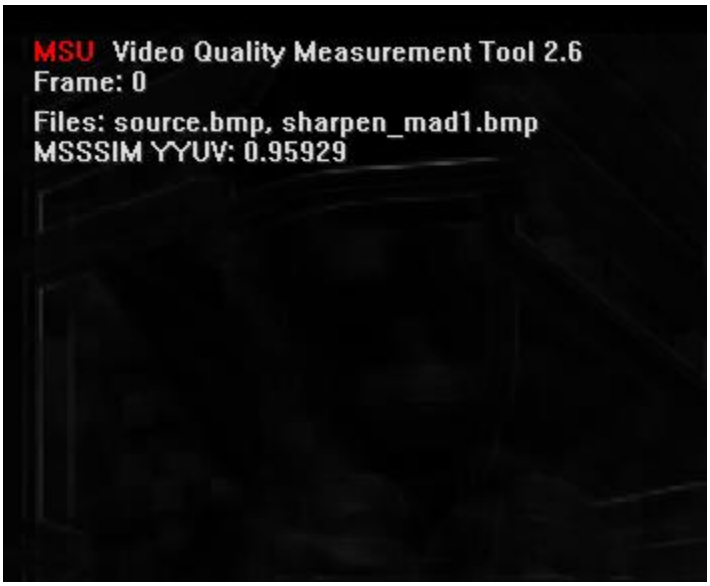
Picture 22. Original and processed video sequences (for MSSSIM example)

Here are the Y-plane MSSSIM (fast) visualizations of these video sequences.

SSIM for image with itself, value = 1

SSIM (fast) for image with noisy image, value = 0.833092

SSIM (fast) for image with blurred image, value = 0.778854



SSIM (fast) for image with sharpen image, value = 0.95929

Picture 23. MSSSIM (fast) visualizations for original and processed video sequences

Here are the Y-plane MSSSIM (precise) visualizations of these video sequences.

SSIM for image with itself, value = 1

SSIM (precise) for image with noisy image, value = 0.905112

SSIM (precise) for image with blurred image, value = 0.812122

SSIM (precise) for image with sharpen image, value = 0.969941

Picture 24. MSSSIM (precise) visualizations for original and processed video sequences

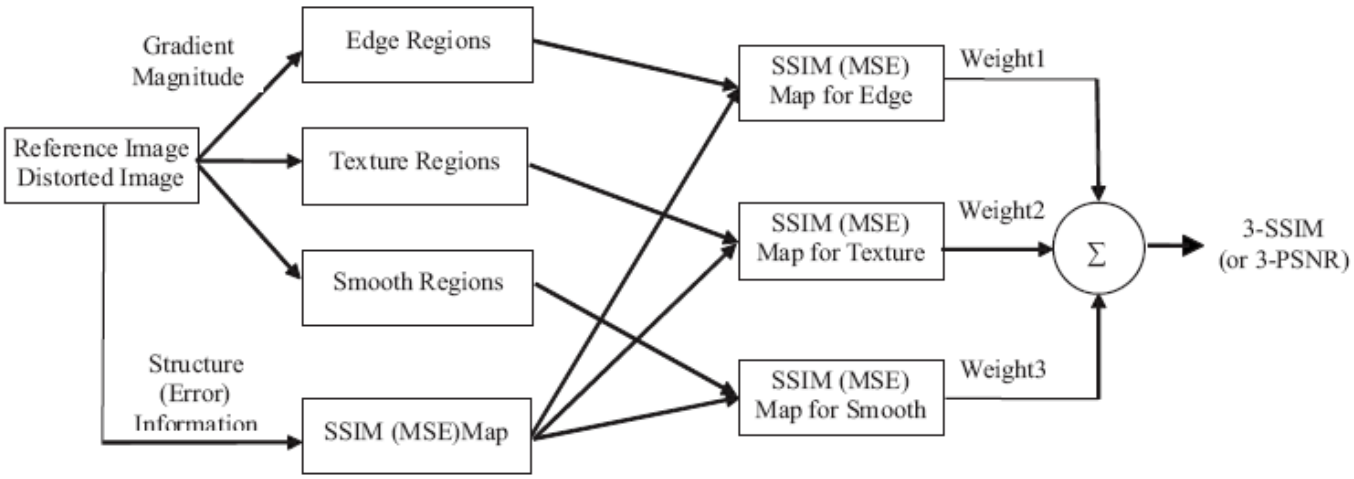
3-Component SSIM INDEX

Brief Description

Original paper is Chaofeng Li, Jiangnan University, School of Information Technology, *"Content-weighted video quality assessment using a three-component image model"*, Journal of Electronic Imaging 19(1), 011003 (Jan–Mar 2010)

This article could be found here:
http://live.ece.utexas.edu/publications/2010/li_jei_jan10.pdf

The scheme of 3-component SSIM calculation could be presented as:



Picture 25. Diagram of the 3-component structural similarity (3SSIM) measurement system[3]

Main idea of the 3-component structure similarity index (3SSIM) is to divide measurement map into 3 regions:

- Edge region
- Texture region
- Smooth region

Regions are determined by calculating gradient magnitude map for both reference and processed images. We determine regions by comparison pixel magnitude value with maximum magnitude value($g_{\max}$). Let TH1 = $(0.12)*g_{\max}$ and TH2 = $(0.06)*g_{\max}$. Denoting the gradient at coordinate (i, j) on the reference image by $p_o(i,j)$ and the gradient on the distorted image as $p_d(i,j)$ pixel classification is carried out according to the following rules:

- if $p_o(x,y) > TH1$ or $p_d(x,y) > TH1$, then the pixel is considered to be an edge pixel.
- if $p_o(x,y) < TH2$ or $p_d(x,y) \le TH1$, then the pixel is regarded as part of a smooth region.
- otherwise, the pixel is regarded as part of a textured region.

[3] This scheme is taken from paper Chaofeng Li, Jiangnan University, School of Information Technology, *"Content-weighted video quality assessment using a three-component image model"*, Journal of Electronic Imaging 19(1), 011003 (Jan–Mar 2010).

Then we measure images using SSIM index and count final measurement as sum of weighted average of each region

$$3SSIM(x,y) = \frac{\alpha * \sum^{Tex} SSIM(x,y) + \beta * \sum^{Edge} SSIM(x,y) + \gamma * \sum^{Smooth} SSIM(x,y)}{width * height}$$

where $SSIM(x,y)$ – similarity index between pixels

and α, β, γ – weights of texture, edges and smooth regions

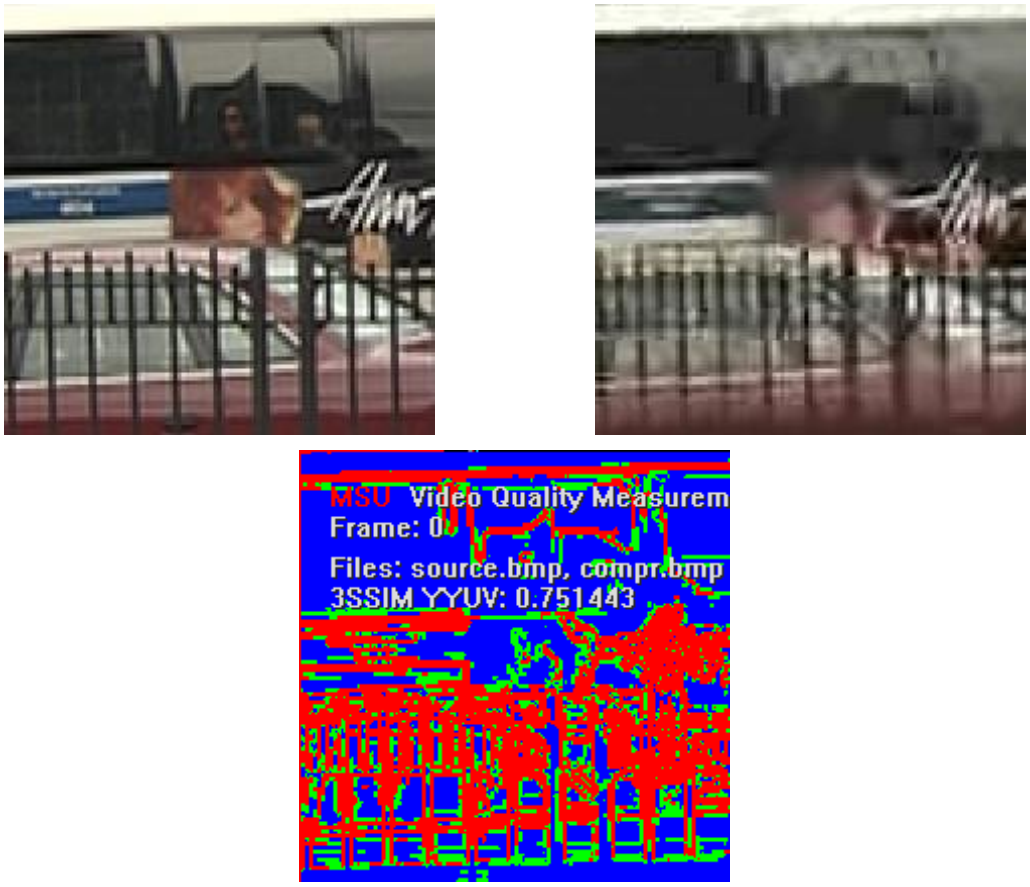
Structure similarity index are calculated using precise SSIM algorithm (using Gaussian window).

In our implementations one MSSSIM value corresponds to two sequences. Values are in range -1…1. Higher values correspond to closer video sequences; value 1 corresponds to equal frames.

The advantage of 3SSIM metric is that it considers that video quality is highly correlated with video content. There are a number of perceptual factors that influence human perception of visual quality. For example, intensity edges certainly contain considerable image information and are perceptually significant. So 3-weighted model is used to approximate human perception.

Examples

Here is an example of 3-component division of images (blue for smooth, green for textures, red for edges):



Picture 26. 3SSIM example of regions division

Here are more examples how different distortions divided by 3-component model.

Original image

Image with added noise

Blurred image

Sharpen image

Picture 27. Original and processed video sequences (for SSIM example)

Here are the regions (same colors as in previous example) :

Regions for image with itself, value = 1

Regions for image with noisy image

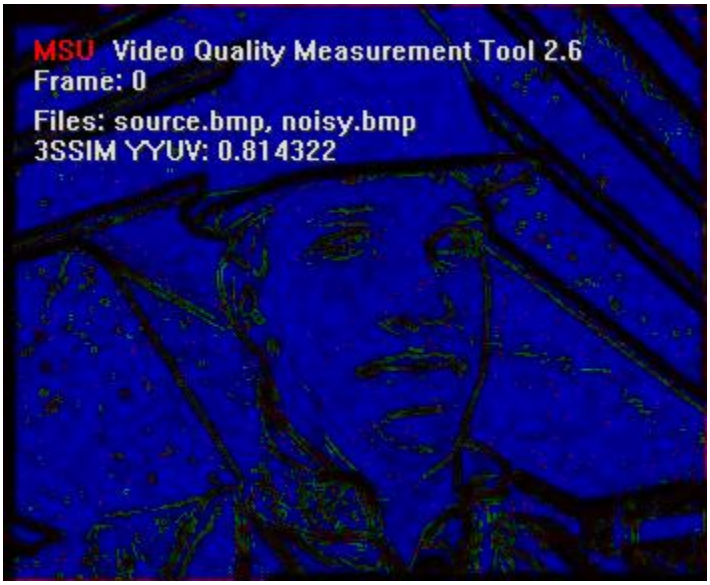
Regions for image with blurred image

Regions for image with sharpen image

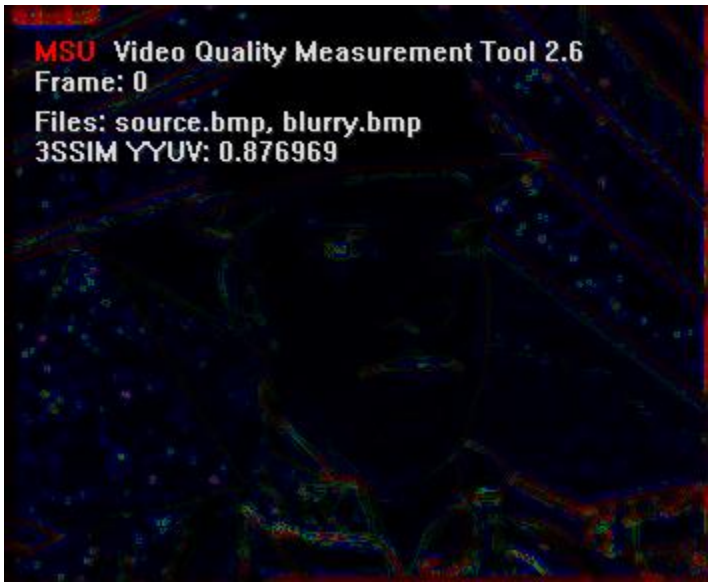
Picture 28. Regions visualization of 3-component SSIM model

Here are the Y-plane 3SSIM visualizations of these video sequences.

SSIM for image with itself, value = 1



3SSIM for image with noisy image, value = 0.814322



3SSIM for image with blurred image, value = 0.876969

3SSIM for image with sharpen image, value = 0.891374

Picture 29. 3SSIM visualizations for original and processed video sequences

Spatio-Temporal SSIM INDEX

NOTE: this metric was excluded from VQMT 11 due to unreliability of results.

Brief Description

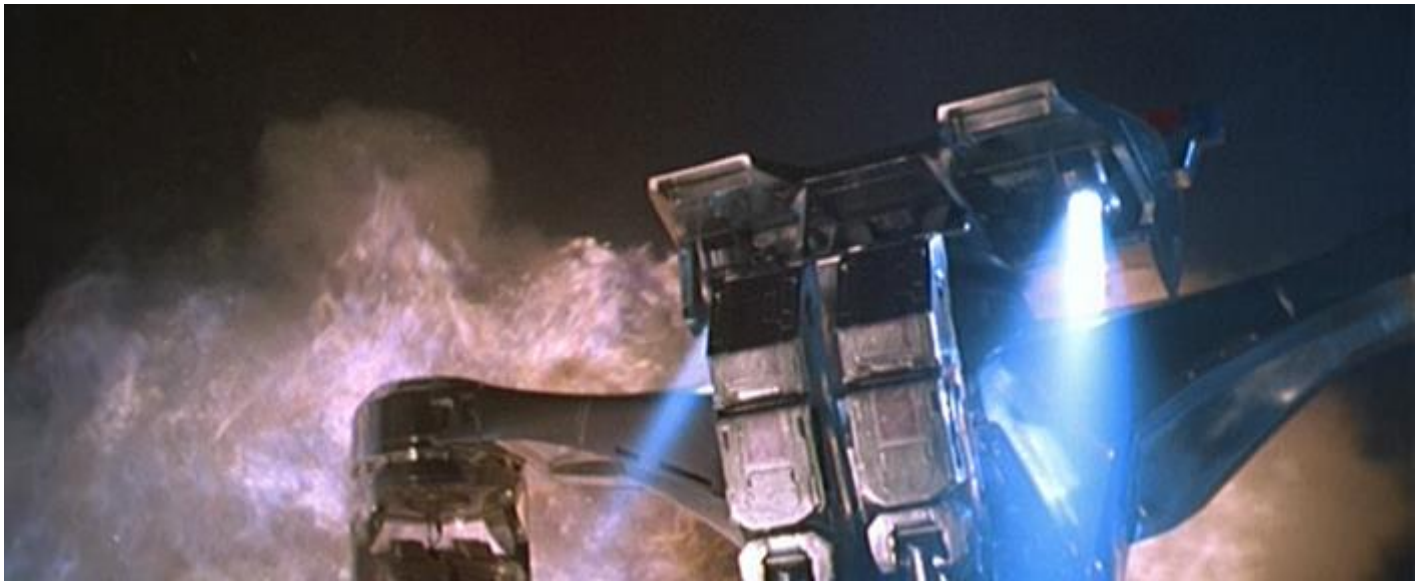
Original paper is Anush K. Moorthy, Alan C. Bovik, Laboratory for Image and Video Engineering (LIVE), Department of Electrical & Computer Engineering, The University of Texas at Austi, “Efficient Motion Weighted Spatio-Temporal Video SSIM Index”, 2010, USA

This article could be found here:
http://live.ece.utexas.edu/publications/2010/moorthy_spie_jan10.pdf

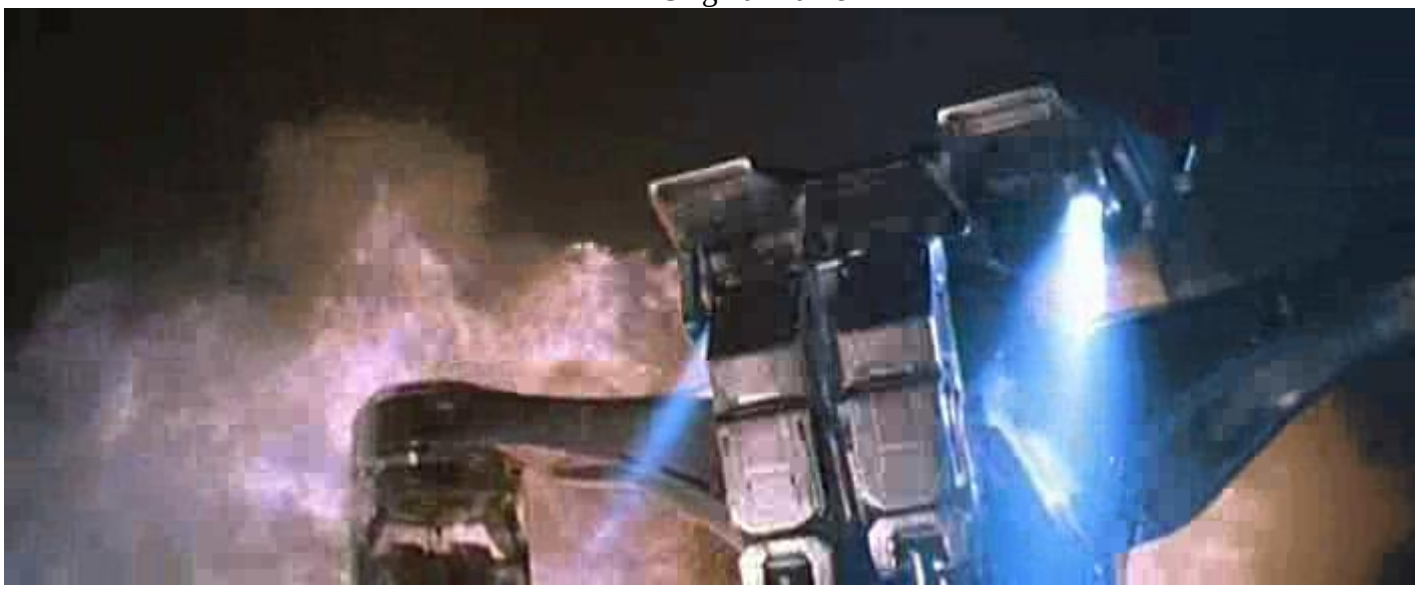
The idea of this algorithm is to use motion-oriented weighted windows for SSIM Index. MSU Motion Estimation algorithm is used to retrieve this information. Based on the ME results, weighting window is constructed for every pixel. This window can use up to 33 consecutive frames (16 + current frame + 16). Then SSIM Index is calculated for every window to take into account temporal distortions as well.

Also another spooling technique is used in this implementation. We use only lower 6% of metric values for the frame to calculate frame metric value. Thus causes larger metric values difference for difference files.

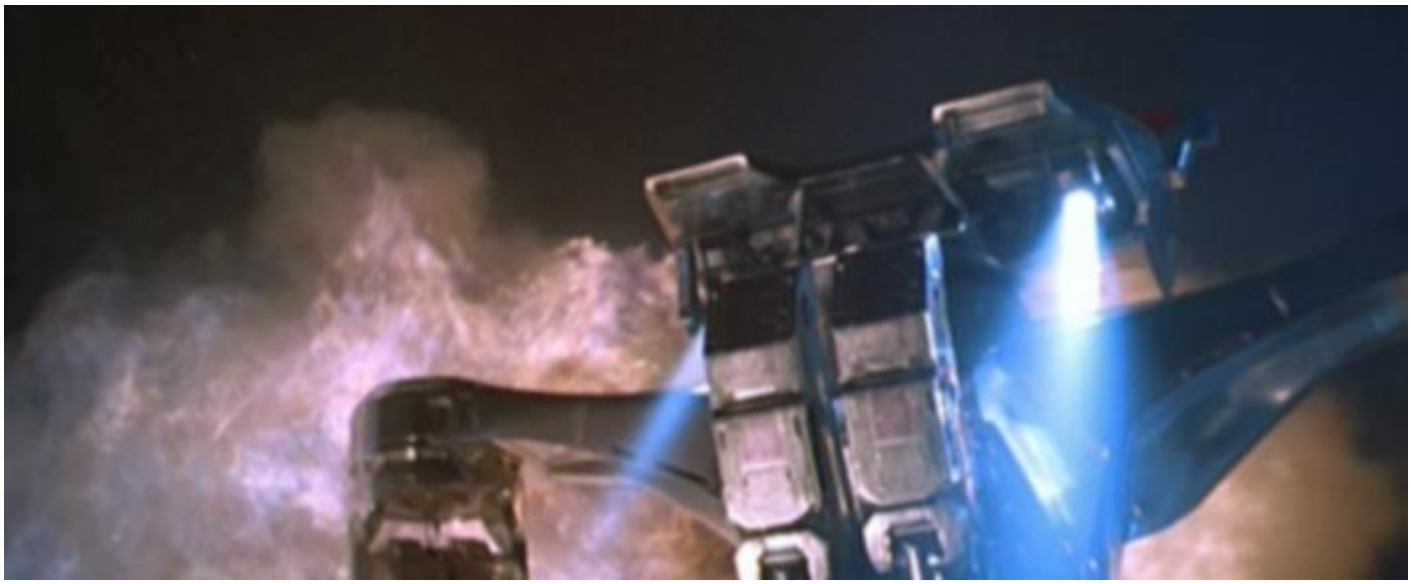
Examples



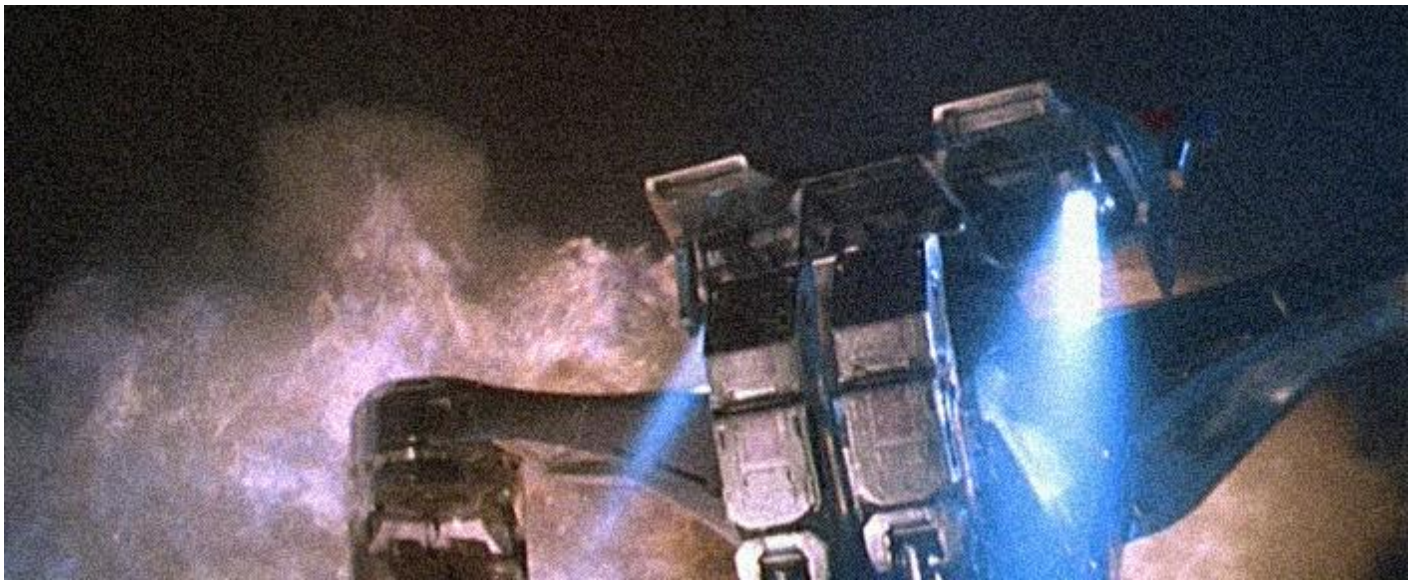
Original frame



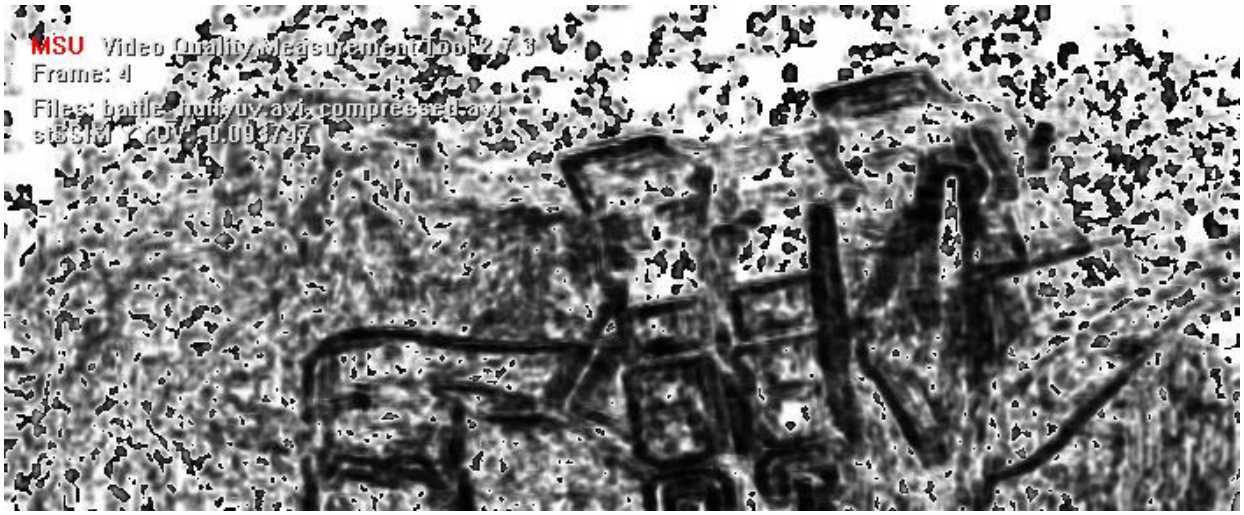
Compressed with XviD frame



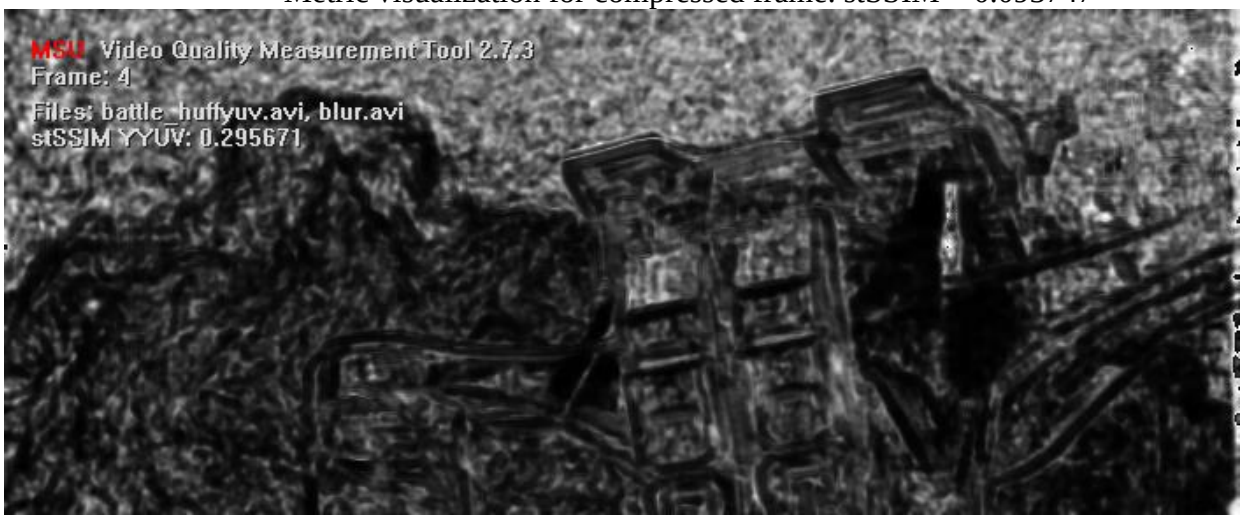
Blurred frame



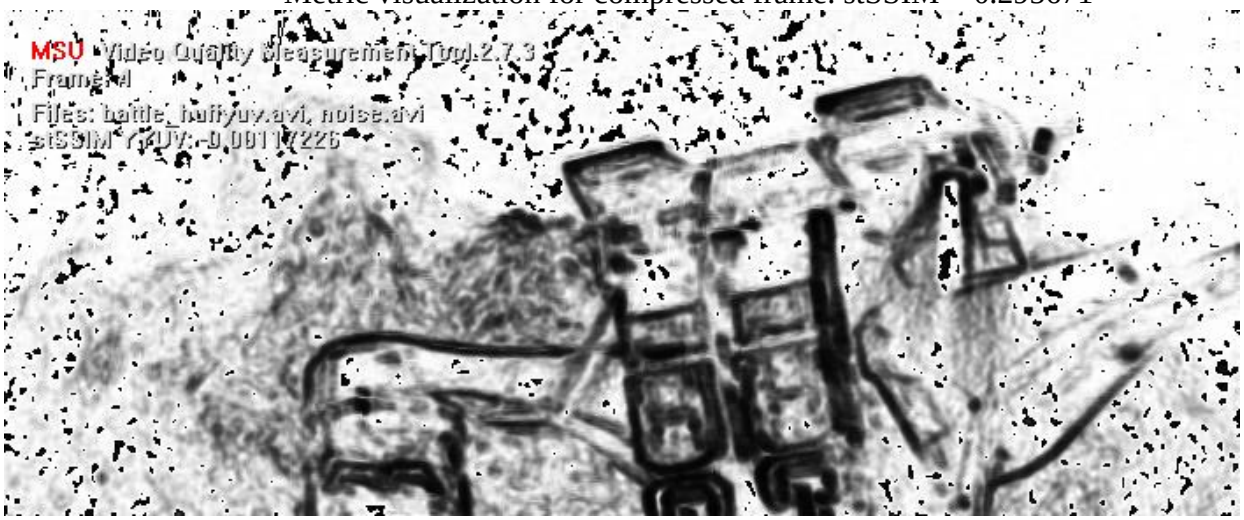
Frame with added noise



Metric visualization for compressed frame. stSSIM = 0.093747



Metric visualization for compressed frame. stSSIM = 0.295671



Metric visualization for compressed frame. stSSIM = -0.00117226

VQM (Video Quality Measure)

Brief Description

Original paper is Feng Xiao, *"DCT-based Video Quality Evaluation"*, Final Project for EE392J

VQM is a DCT-based video quality metric. Following calculations are processed to get value of metric:

- **Color transform.** YUV color space is used for metric calculation.
- **DCT transform** of blocks 8x8. It is used to separate images into different frequencies.
- **Conversion** from DCT coefficients to local contrast (LC) using following equation:

$$LC_{i,j} = DCT_{i,j} \cdot \frac{(DC/1024)^2}{DC},$$

  where DC is the DCT coefficient with indexes (0, 0).

- **Conversion** from LC to just-noticeable difference:

$$JND_{i,j} = LC_{i,j} \cdot CSF_{i,j},$$

  where CSF is Contrast Sensitivity Function. Inverse MPEG-4 default quantization matrix is used as CSF in original article.

- **Weighted pooling of mean and maximum distortions**. First, absolute difference "*D*" is calculated for JND coefficients following by VQM value construction:

$$VQM = mean(|D|) + 0.005 \cdot \max(|D|)$$

This metric uses DCT to correspond to human's perception. Comparing to MSE-based metrics, its performance is much better in these situations when RMSE fails.

Values is greater than 0. One value for two sequences. 0 for equal frames, lower values are better.

Examples

Here is an example of VQM result visualization for original and processed images.

Original Processed VQM

Picture 30. VQM example for processed image

Here are more examples how different distortions have influence on VQM value.

Original image

Image with added noise

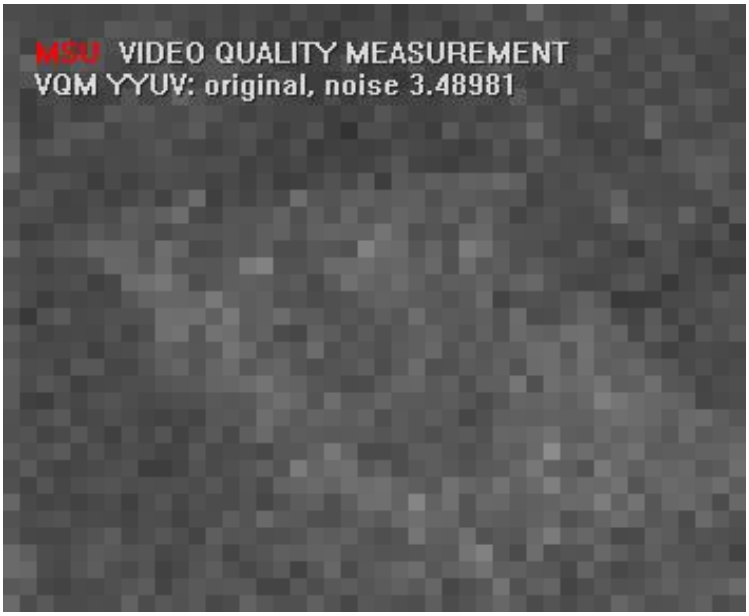
Blurred image

Sharpen image

Picture 31. Original and processed images (for VQM example)

And here are the VQM values of Y-plane for these images

VQM for image with itself, value = 0



VQM for image with noisy image, value = 3.48981

VQM for image with blurred image, value = 1.63067

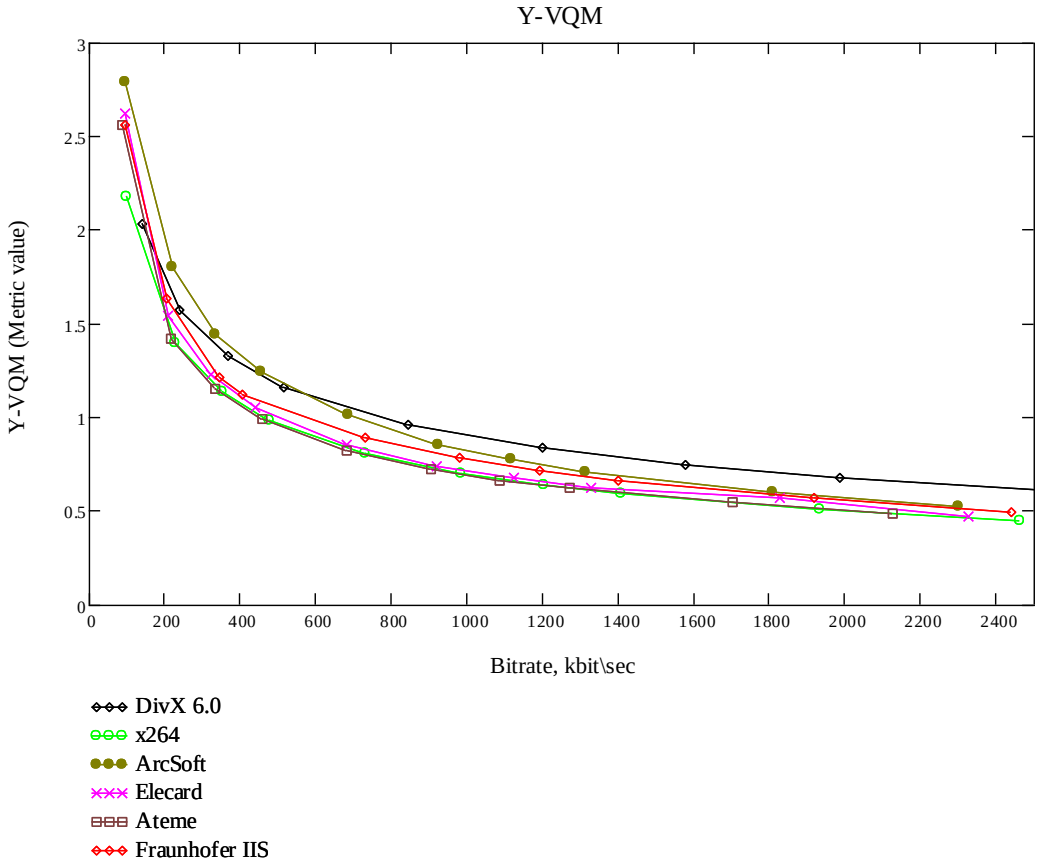
VQM for image with sharpen image, value = 1.31699

Picture 32. VQM values for original and processed images (for VQM example)

Also MSU VQMT with VQM metric was widely used during MSU Codec Comparisons, including Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison

http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2005_en.html

Here is an example of RD-curve from Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison:



Picture 33. VQM Example for RD-curve

MSU Blurring Metric

Brief Description

Blurring effect is one of video compression artifacts. The main source of this artifact is transform coefficients quantization during encoding. High-frequency component of information suffers during this process in the first place. In spite of low perceptibility of HVS to high-frequency band, such artifacts are often visible to video viewers. Another source of blurring effect is deblocking algorithms. Trying to smooth colors along the block border, these algorithms can smooth some object borders because of algorithms mistakes. This leads to damaging of important, critical for HVS border information.

This metric allows you to compare power of blurring of two images. If value of the metric for first picture is greater than for second, it means that second picture is more blurred, than first.

One value for one video sequence.

Main features: this metric is fast and doesn't require source video.

This method estimates color variance in the neighborhood of a pixel and computes average variance. This metric has 2 variations:

- Sigma (default since VQMT 11). It uses 3-pixel radius neighborhood and normalized Gaussian kernel. Range of values for this modification is fixed: 0(totally smoothed)..1(very noisy)
- Delta (the only before VQMT 11). It uses 1-pixel radius neighborhood. This method does not define particular range: the bigger value means more noisy (for substantially same images of similar size).

NOTES:

1. You can't measure blurriness on constant or gradient areas of input image. So, the value of metric is very dependent on amount of edges in images.
2. This metric will detect not only compression artifacts, but natural not-in-focus areas, so the value of metric is very dependent on area of focused objects in the frame.

The delta method to estimate picture smoothness is calculation of brightness change in the neighborhood of current pixel. Considering video frame as continuous function I(x,y), one can calculate function gradient:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

Magnitude of brightness change can be estimated as magnitude of gradient:

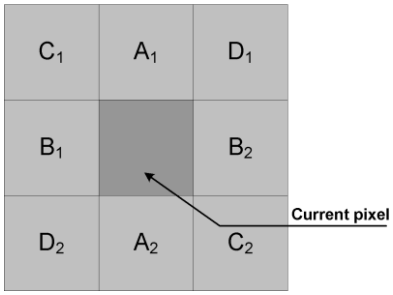
$$V = |\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x} \right)^2 + \left(\frac{\partial I}{\partial y} \right)^2}$$

Difference derivations should be used instead of exact solution in case of discrete picture. We used central difference derivation. Formula below shows approximation of partial X derivative:

$$\frac{\partial I}{\partial x} \approx \frac{I(x-1,y)+I(x+1,y)}{2}$$

Additionally, following formula were used to approximate gradient magnitude:

$$|\nabla I| = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$$



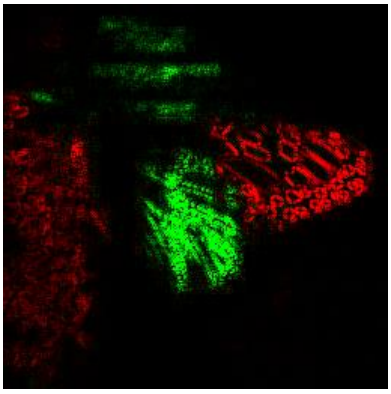
Picture 34. Used for blurring metrics calculation pixels

Such approximation allows to avoid complex operation of square root calculation and doesn't decrease precision significantly. As a result only four pixels, three adding and two modulus operation are used to calculate metric value for each pixel (see Picture 34):

$$V_{blurring} = abs(A_1 - A_2) + abs(B_1 - B_2)$$

A

B



C

Picture 35. Examples of blurring metrics

A. Original frame. B. Processed frame with blurring and contrast changing. C. Gradient magnitude estimation.

Examples

Here are examples how different distortions (like blurring and sharpening) have influence on MSU Blurring value.

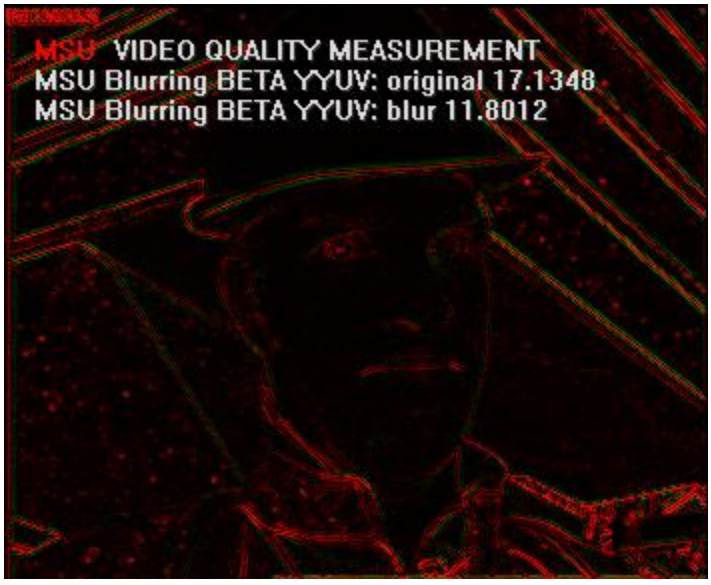
Original image

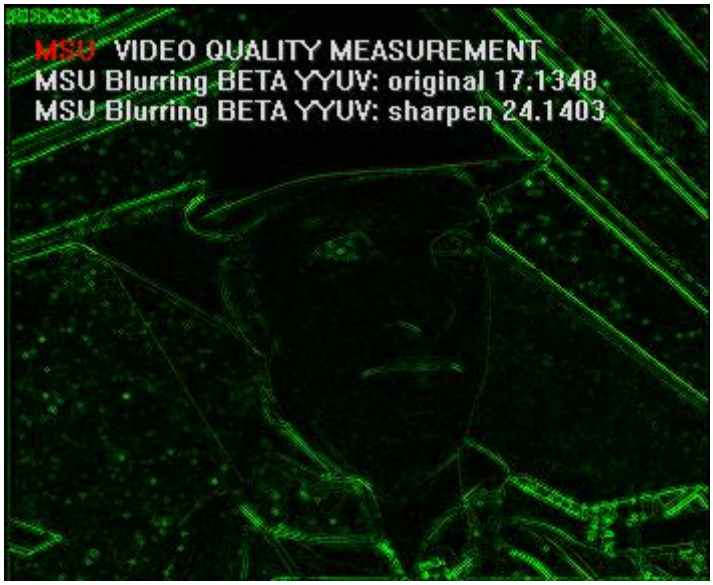
Blurred image

Sharpen image

Picture 36. Original and processed images (for MSU Blurring example)

And here are the MSU Blurring values of Y-plane for these images





MSU Blurring for image with blurred image, values

- 17.1348 for original image
- 11.8012 for blurred image

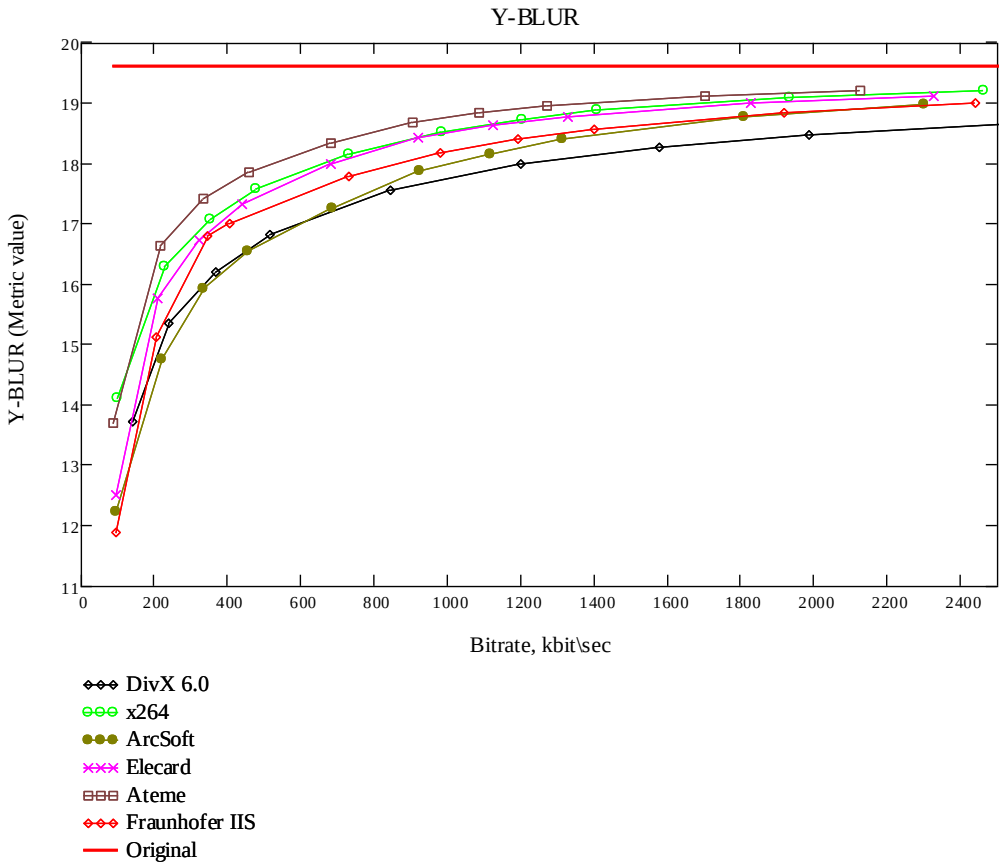
MSU Blurring for image with sharpen image, values

- 17.1348 for original image
- 24.1403 for sharpen image

Picture 37. MSU Blurring values for original and processed images

Also MSU VQMT with PSNR metric was widely used during MSU Codec Comparisons, including Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2005_en.html

Here is an example MSU Blurring graph from Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison:



Picture 38. MSU Blurring example

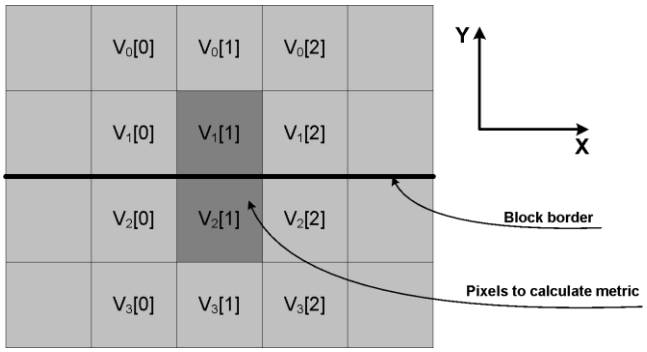
MSU Blocking Metric

Brief Description

Most modern algorithms of video compression including MPEG-2, MPEG-4 ASP, H.263, MPEG-4 AVC/H.264 and some others divide each frame into blocks of predefined size. Motion compensation technique is applied to each block after transform of estimated residual. The purpose of transform is to reduce dependencies between block's pixels. Resulting coefficients are quantizing and coding using lossless compression. Information loss during quantization produces number of artifacts in compressed video such as blocking effect, blurring effect, Gibbs effect, etc.

Blocking effect appears because of separate blocks transformation. Adjacent blocks distort independently, resulting in big brightness differential at the blocks boundaries in decoded sequences. This effect becomes stronger simultaneously with increasing quantize coefficient (decreasing information after quantization). Visibility of blocking artifact is additionally connected with features of HVS. It is well known that high-frequency artifacts (including blocking) are better visible in smooth areas than in high-detailed areas. This HVS feature was taken into account in metric's algorithm with the help of area contrast estimation.

This metric also contains heuristic method for detecting objects edges, which are placed to the edge of the block. In this case metric value is pulled down, allowing to measure blocking more precisely. We use information from previous frames to achieve better accuracy.

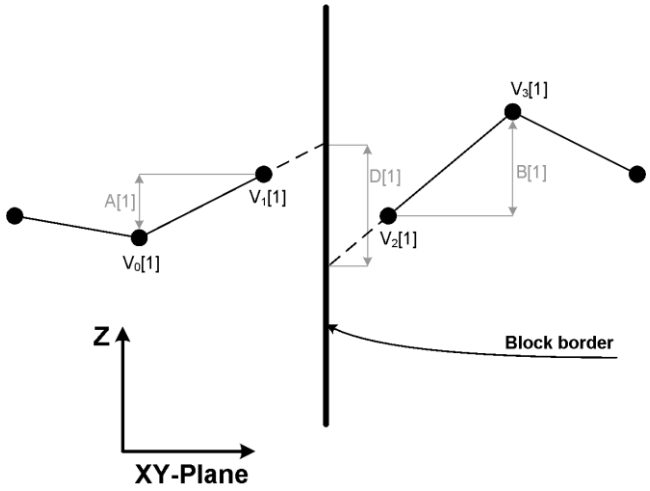


Picture 39. Pixels, used for blocking metrics calculation

Metric is calculated for pixels at boundaries of 8x8 blocks. The metric value is the same for each two adjacent to blocks boundary pixels (dark gray pixels at Picture 39). That value depends on two factors: magnitude of color difference at block's boundary and picture contrast near boundaries. The former is calculated using the following expressions:

$$A = V_0 - V_1$$
$$B = V_1 - V_2$$
$$C = V_2 - V_3$$
$$D = C - \frac{A + B}{2}$$
$$M = abs(D[1] + D[2] + D[3])$$

Consider lines produced by values of two pixels from each side of block boundary. Each component of vector D is the difference between prolongations of these lines to block boundary (Picture 40). So, geometric sense of vector D is the magnitude of color difference at block's boundary.

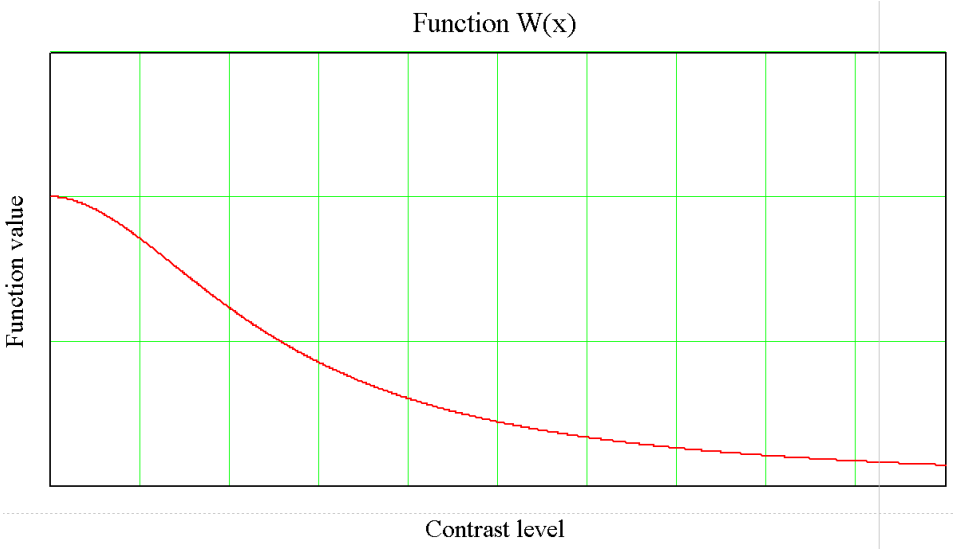


Picture 40. Geometric sense of D for MSU Blocking

Contrast near block's boundary is calculated using the following formulas:

$$W_1 = W(abs(A[0]) + abs(B[0]))$$
$$W_2 = W(abs(A[1]) + abs(B[1]))$$
$$W_3 = W(abs(A[2]) + abs(B[2]))$$
$$W_R = (W_1 + W_3) \cdot W_2$$

The higher contrast value the lower is a contrast coefficient W_R. Such coefficient behavior achieved with the help of shape of function W(x). Important feature of this function is slow decreasing speed at low values of argument. Contrast coefficient is near one in smooth areas and doesn't influence on resulting metric's value. On the other hand, contrast coefficient is low for contrast areas, which decrease resulting metrics value.



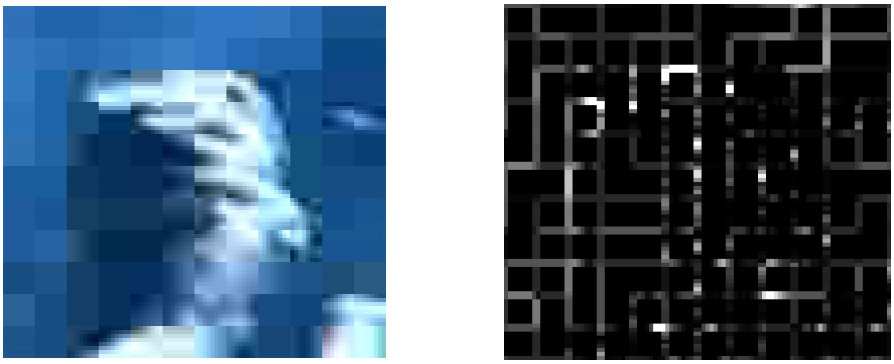
Picture 41. Shape of function W(x) for MSU Blocking

Resulting metric's value $V_{blocking}$ can be obtained by multiplying color break value M and contrast coefficient W_R:

$$V_{blocking} = M \cdot W_R$$

Examples

Example of blocking metric visualization is shown at Picture 42.



A **B**

Picture 42. A. Decoded frame. B. Visualization of blocking metric

Here are examples how compression has influence on MSU Blocking value.

Original image Compressed image (JPEG)

Picture 43. Original and compressed images (for MSU Blocking example)

And here are the MSU Blurring values of Y-plane for these images

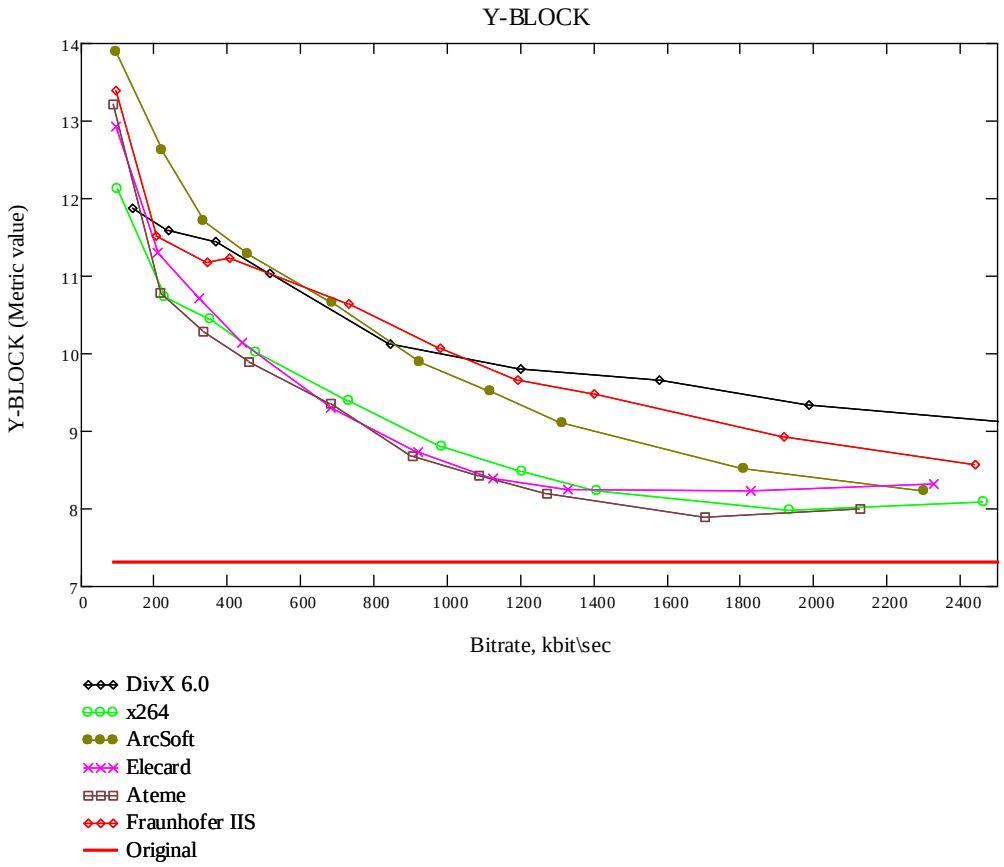
MSU Blocking for image with blurred image, value is 7.7239

MSU Blocking for image with sharpen image, value is 38.3087

Picture 44. MSU Blocking values for original and processed images

Also MSU VQMT with PSNR metric was widely used during MSU Codec Comparisons, including Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2005_en.html

Here is an example of MSU Blocking graph from Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison:



Picture 45. MSU Blocking example

NIQE Metric

Brief Description

This is the first metric in VQMT that can perform no-reference measure of video/image quality (naturalness). This metric described in Mittal, A., R. Soundararajan, and A. C. Bovik. "Making a Completely Blind Image Quality Analyzer." IEEE Signal Processing Letters. Vol. 22, Number 3, March 2013, pp. 209–212.

This metric uses Machine Learning, so correct visualization is not possible for this metric. Visualization currently not implemented for NIQE.

NOTE: this metric only applicable to filmed scene. Metric can produce inadequate result in case of rendered sequences (including credits, subtitles, graphic, etc.). Please, run NIQE excluding all rendered scenes, they can fatally spoil average value.

NOTE: this metric can produce bad result on scenes containing noisy objects, like sand or grass, however on scenes with big constant areas, like monotonic sky. In common case normal metric results lies in the interval 1..15. VQMT filter bigger values by-default. You can configure filtering with metric configuration.

NOTE: sometimes, metric shows better result for the compressed image and this correlates with human perception. Compressed image not always is perceived as worse. It can occur for example in case of noisy images (if the noise has non-compression nature). This is only metric in VQMT now that can detect increasing of subjective quality in comparison to original.

NOTE: sometimes, codec can allow geometry transformation (like shift of heterogeneous objects in frame), that not critical for subjective perception. Objective-reference metrics are very perceptive to such transformation, and in this cases no-reference metric can show result closer to subjective score.

Configuration

- **Mean threshold**
  Values of metric greater than this value will be skipped during mean calculation. 0 for disable skipping
  Default value: 15
  Usage: -set "mean_thresh=<value>", where <value> can be:
    - any floating point number

- **Threshold smoothing**
  Values of metric greater that 'Mean threshold' + 'Threshold smoothing' will be skipped, values less than 'Mean threshold' - 'Threshold smoothing' will be assumed with weight 1. Intermediate values will be taken with intermediate weight
  Default value: 5
  Usage: -set "mean_thresh_smoothing=<value>", where <value> can be:

- any floating point number

- **Type of normalization**
  Default value: "native"
  Usage: -set "norm_alg=<value>", where <value> can be:
  - fast - the fastest algorithm, low precision
  - native - like in native NIQE implementation. Slowest one
  - precise - the most precise algorithm

VMAF Metric

Brief Description

VMAF metric is the most modern reference metric that combines different algorithm in it and using Machine Learning principles to operate. Generally, VMAF is not a metric, it is a technology. To calculate VMAF you should specify trained model.

Before model applied, VMAF calculates basic features of the next groups:

- adm (17 features)
- ansnr (2 features)
- vif (16 features)
- motion (2 features)

Than it applies SVM model to feature values to compute destination result.

VMAF is video metric, and it's value has temporal dependency. Actually, value of VMAF depends on previous and next frame (feature motion from group motion depends from previous frame, feature motion2 depend from next and previous frame)

In VQMT you can compute elementary features (1 feature from each group), common features (features, that used in build-in models), or all features.

Since, metric uses Machine Learning, it's impossible to make clear visualization, so you can choose one of 4 algorithms to visualize.

Models

VMAF has several built-in models, trained by Netflix. However, it can use your own model from file. To use your own model, you should have at least 2 files: .pkl and .model.

NOTE: each model trained for particular resolution and particular screen size/distance from screen ratio.

Built-in models of VMAF:

- **VMAF v0.6.1**. Default if not using bootstrap model and not 4k resolution
- **VMAF v0.6.1 4k**. Default for 4k resolution
- **VMAF v0.6.2**. Supports bootstrap
- **VMAF v0.6.2 4k**. Supports bootstrap. Default bootstrap model for 4k
- **VMAF v0.6.3**. Default bootstrap model for non-4k.
- **VMAF v0.6.0**. Obsolete model

NOTE: VQMT can automatically choose 2k or 4k variation for built-in model (if exists). If you are using custom model, you should be sure, that you are doing right.

Bootstrap models and confidence intervals

VMAF can produce statistical result (if model supports). For this, you should have bootstrap model, which is actually not a single model, but a bundle of models. VMAF will calculate result of each model and then use obtained values for statistical information: confidence interval, standard deviation, mean.

You also can see values of all models if you turn on Per-model values option.

Configuration

- **Model preset**
  Choose built-in model or 'custom' for loading model from file. Built-in models:
  Default value: "default"
  Usage: -set "model_preset=<value>", where <value> can be:
  - default - VMAF default behaviour:
    - VMAF v0.6.1 for running without confidence interval and per-model values
    - VMAF v0.6.1 4k for previous case if applying 4k model
    - VMAF v0.6.3 for running with confidence interval or per-model values
    - VMAF v0.6.2 4k for previous case if applying 4k model (NOTE: no v0.6.3 for 4k)
  - vmaf_v061 - Netflix model VMAF v0.6.1 (2k or 4k)
  - vmaf_v062 - Netflix model VMAF v0.6.2 (2k or 4k), supports confidence interval
  - vmaf_v063 - Netflix model VMAF v0.6.3 (only 2k), supports confidence interval
  - vmaf_v060 - Netflix model VMAF v0.6.0 (only 2k)
  - basic_features - view only basic features from VMAF. Model will not be applied
  - standard_features - features that is used in VMAF v0.6.1 and VMAF score (2k or 4k)
  - all_features - view all features from VMAF. Model will not be applied
  - all - all feature and next models:
    - VMAF v0.6.1 (2k or 4k)
    - VMAF v0.6.2 (2k or 4k)
    - VMAF v0.6.3
  - custom - specify file

- **Custom model (*.pkl)**
  you can specify path to *.pkl file here (or multiple ;-separated *.pkl files). Model file should be placed near pkl file.
  NOTE: this only means if preset is set to 'custom'
  Default value: ""
  Usage: -set "custom_model_files=<value>", where <value> can be:
  - any string

- **4k**
  selection 4k model policy:
  NOTE: this param not affects custom model
  Default value: "auto"
  Usage: -set "4k=<value>", where <value> can be:
  - auto - select 4k if exists suitable model and input video is 4k
  - forced_2k - always 2k model
  - forced_4k - 4k if exsists: VMAF v0.6.1-2

- **Confidence interval**
  turn on additional VMAF features: 95%-confidence interval output and other statistical information
  Default value: false
  Usage: -set "confidence_interval=<value>", where <value> can be:
    - true
    - false

- **Per-model values**
  output values for all bootstrap models if confidence interval is on
  Default value: false
  Usage: -set "permodel_values=<value>", where <value> can be:
    - true
    - false

- **Visualize algorithm (if on)**
  if visualization turned on you can select feature to visualize. It's impossible to calculate distribution of real VMAF value, so you can only visualize one of supposed features
  Default value: "adm"
  Usage: -set "visualize_alg=<value>", where <value> can be:
    - adm
    - ansnr
    - motion
    - vif

- **Use phone model**
  turn on postprocessing of metric value that produces more precise results for handheld devices. Select 'both' to see both results with and without postprocessing
  Default value: "no"
  Usage: -set "phone_model=<value>", where <value> can be:
    - no
    - yes
    - both

- **Disable clipping values**
  turn off clipping value to range set by model (0..100 for example)
  Default value: false
  Usage: -set "disable_clip=<value>", where <value> can be:
    - true
    - false

- **Use multithreading**
  you can turn off multithreading of this metric, in some cases, it can increase speed by doing parallelization of other tasks
  Default value: true
  Usage: -set "multithreaded=<value>", where <value> can be:
    - true
    - false

SI metric

Brief Description

This is no-reference metric, that calculates Spatial perceptual information. This metric measures complexity (entropy) of an input image. This metric represents simplest SI realization that takes standard deviation of sequence of pixel values (Y-component) of Sobel transformation of input image

$$SI(X) = std[Sobel(x(i, j))] / \max$$

The values are in 0..1. 0 – for simple (monotone) frame, 1 – for very complex frame.

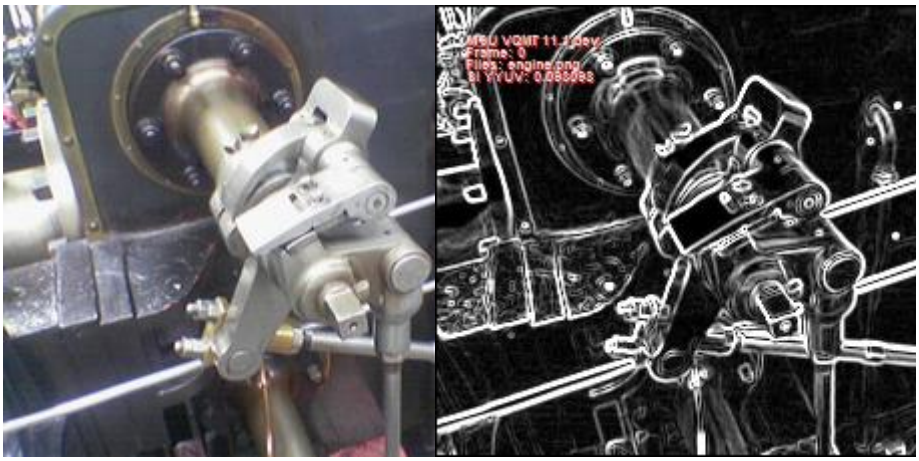
Sobel is length of vector ($Sobel_h$, $Sobel_v$), where $Sobel_h$ and $Sobel_v$ are horizontal and vertical Sobel transformation. While calculating Sobel, the edge pixels will be excluded from calculation.

Visualization of this metric will show Sobel transformation of input image.

For more information please see *ITU-T Recommendation P.910: Subjective video quality assessment methods for multimedia applications, 1999. – 37 p.*

Examples

Here is example of this metric



Input image SI visualization

Picture 46. SI sample visualization

TI metric

Brief Description

This is no-reference metric, that calculates Temporal perceptual information. This metric measures complexity (entropy) of difference between consequent frames of input video. This metric represents simplest TI realization that takes standard deviation of sequence of differences of corresponding pixel values of a frame and previous frame:

$$TI(V) = std[V_n(x, y) - V_{n-1}(x, y)] / \max$$

The values are in 0..1. 0 – for simple (static) video, 1 – for video with very diverse frames. Also, value 0 for the first frame.

Visualization of this metric will show difference between frame and previous frame.

For more information please see *ITU-T Recommendation P.910: Subjective video quality assessment methods for multimedia applications, 1999. – 37 p.*

Examples

Here is example of this metric

A frame Next frame TI visualization

Picture 47. TI example for processed image

Metrics GPU acceleration

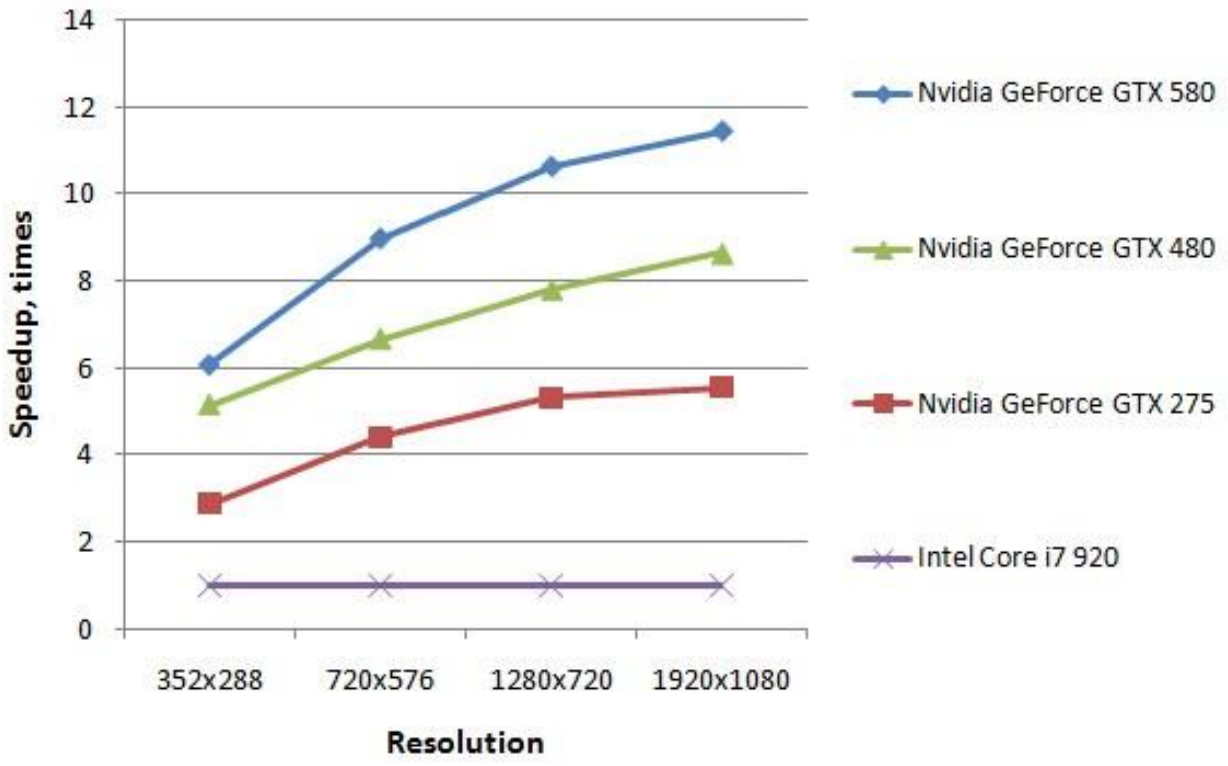
Brief Description

Now SSIM-based algorithms show best subjective quality correlation among other video quality algorithms. To increase performance of SSIM-based metrics, these algorithms were implemented on graphics and other computing hardware. You can choose between CUDA technology (suitable only for NVidia Graphical Cards) or OpenCL. SSIM, 3-SSIM, and MS-SSIM metrics implemented for now.

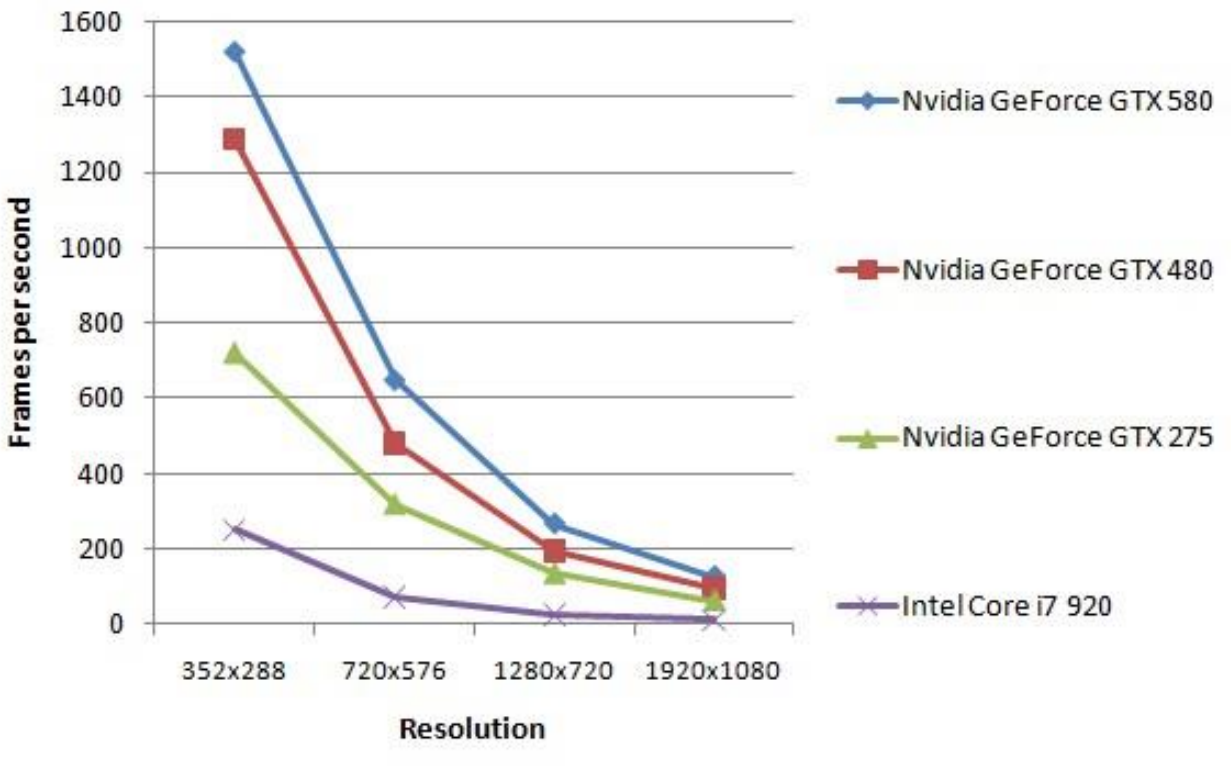
NOTE: OpenCL is recommended. Metrics has faster implementation for OpenCL and demands less resources. For Linux only OpenCL GPU metrics are available.

These implementations can be found in the metric list in the GUI or via **-metr ssim_cuda, 3ssim_cuda, msssim_cuda** parameters via console line interface.

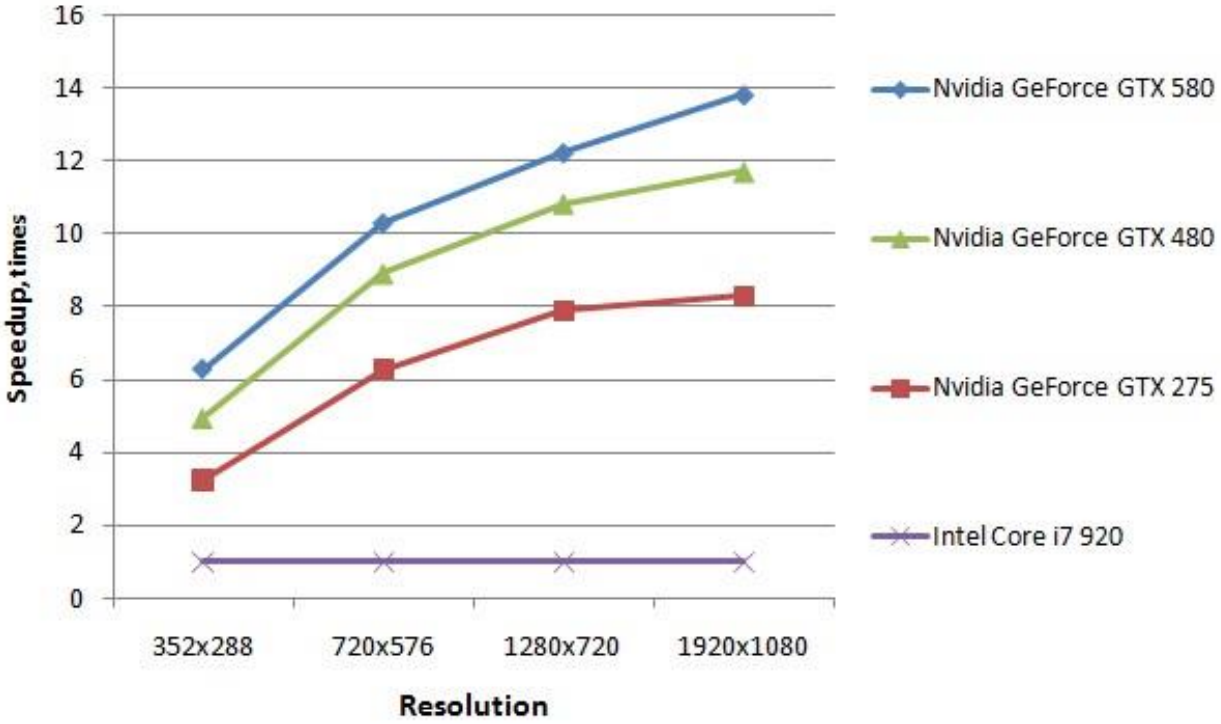
Speedup results provided in the graphs below as fps graph and speedup graphs:



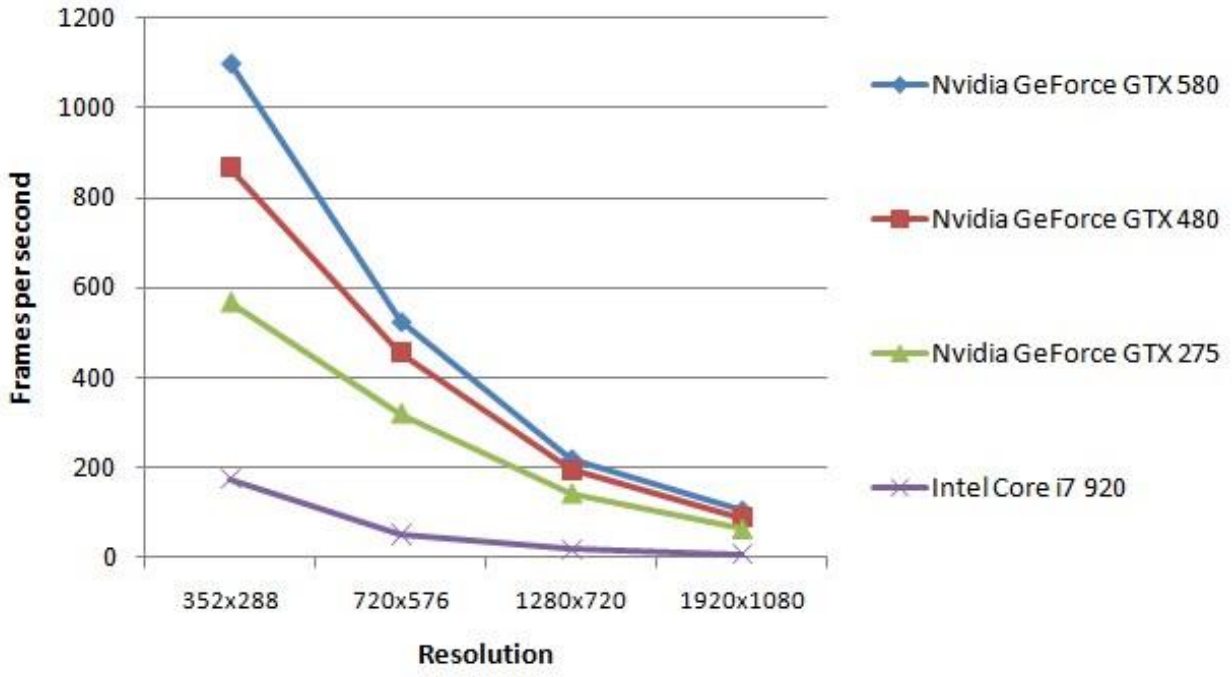
Picture 48. SSIM metric speedup



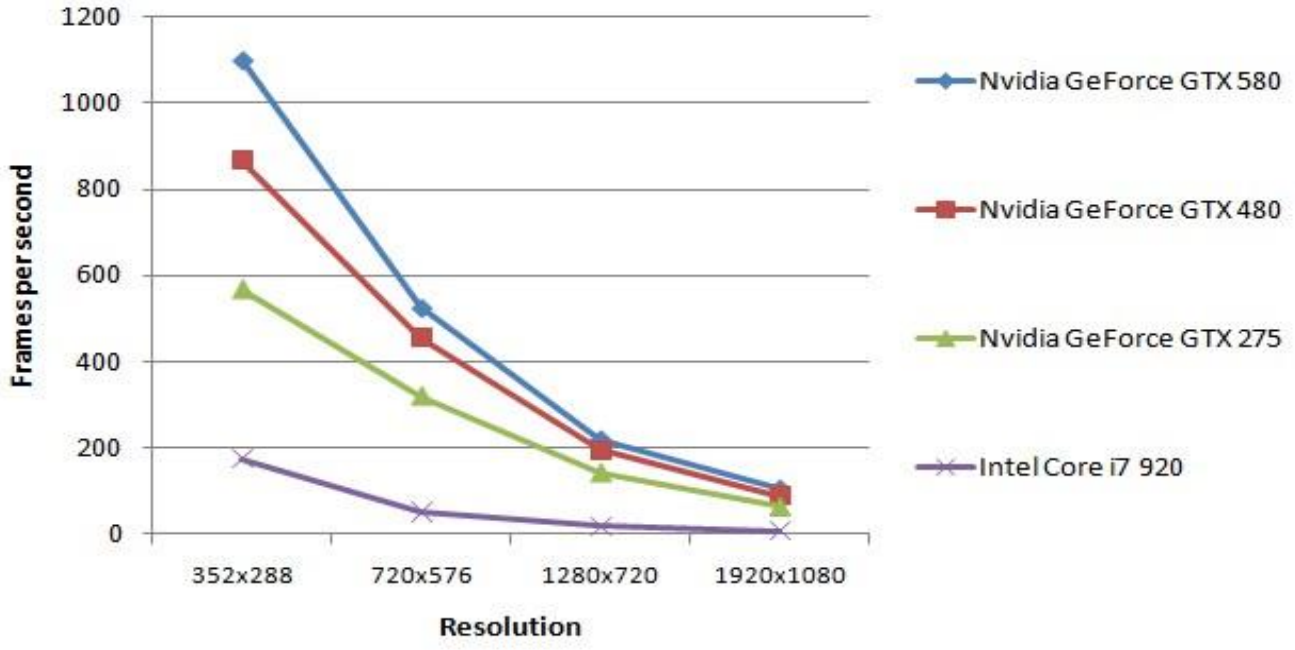
Picture 49. SSIM metric fps graph



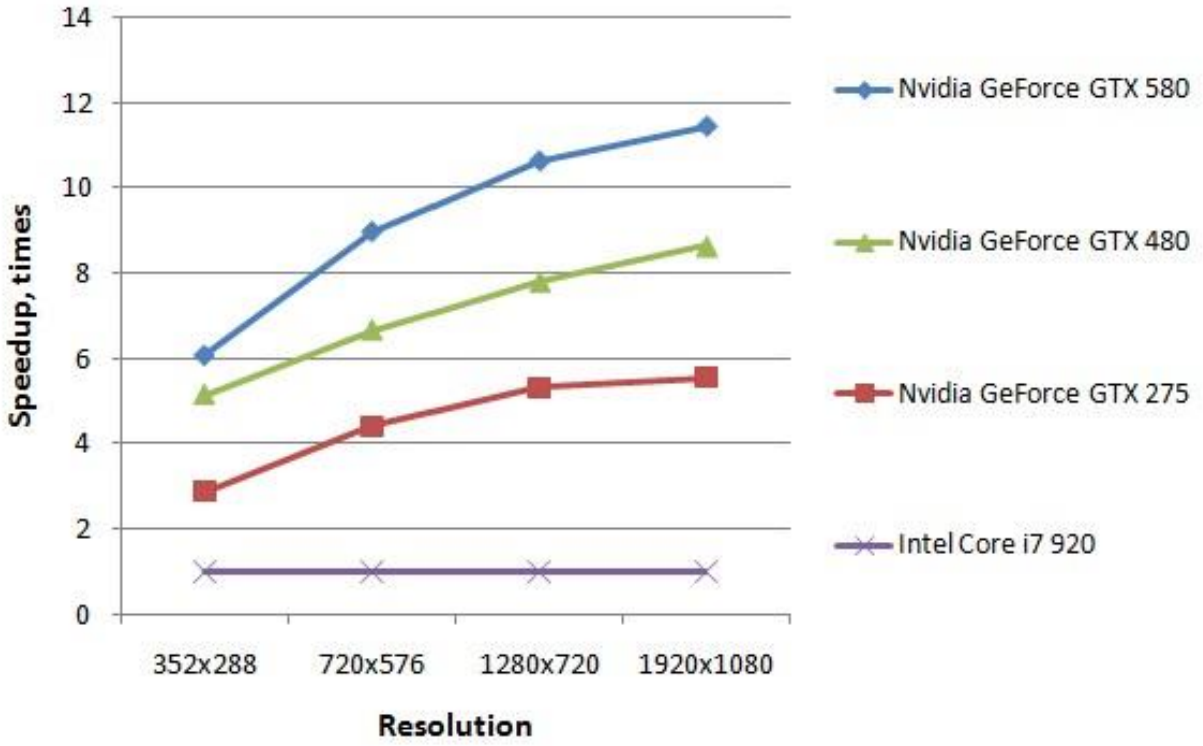
Picture 50. MSSSIM metric speedup



Picture 51. 3SSIM metric fps graph



Picture 52. MSSSIM metric speedup



Picture 53. MSSSIM fps metric graph

Subjective quality metric comparison

There are a lot of full reference objective quality metrics and each of them somehow represents difference between two video sequences. There are comparisons with subjective quality for each metrics can be performed to understand which metric is better. We provide such comparison for the most popular metrics: PSNR, SSIM, 3-SSIM, MS-SSIM and new stSSIM. We provide two indices of similarity between subjective quality and objective metric values for these metrics: Spearmen Rank Order Correlation Coefficient (SROCC) and Pearson Linear Correlation Coefficient (LCC). Coefficients provided for public video bases with subjective quality values available: Laboratory for Image & Video Engineering Video Quality Database (http://live.ece.utexas.edu/research/quality/live_video.html) and Video Quality Experts Group Phase I video sequences database (http://www.its.bldrdoc.gov/vqeg/downloads/downloads.php).

Spearman rank order correlation coefficient

Spearman Rank Order Correlation Coefficient is a non-parametric measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic function. It is defined as the Pearson correlation coefficient between the ranked variables. Raw scores X_i, Y_i are converted to ranks x_i, y_i and coefficient computes as:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}.$$

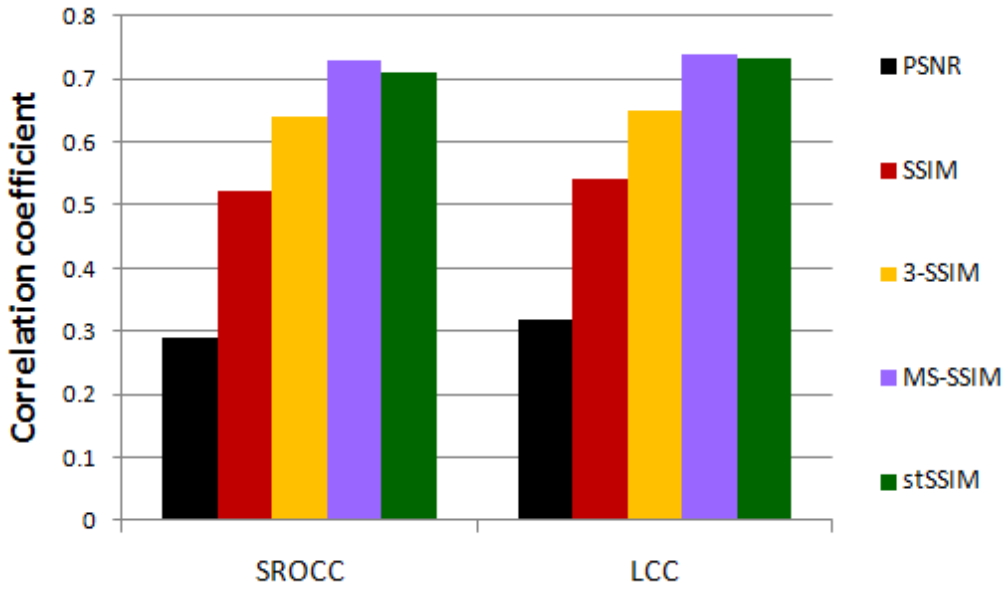
More information about SROCC you can find on the Wikipedia page http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient.

Pearson linear correlation coefficient

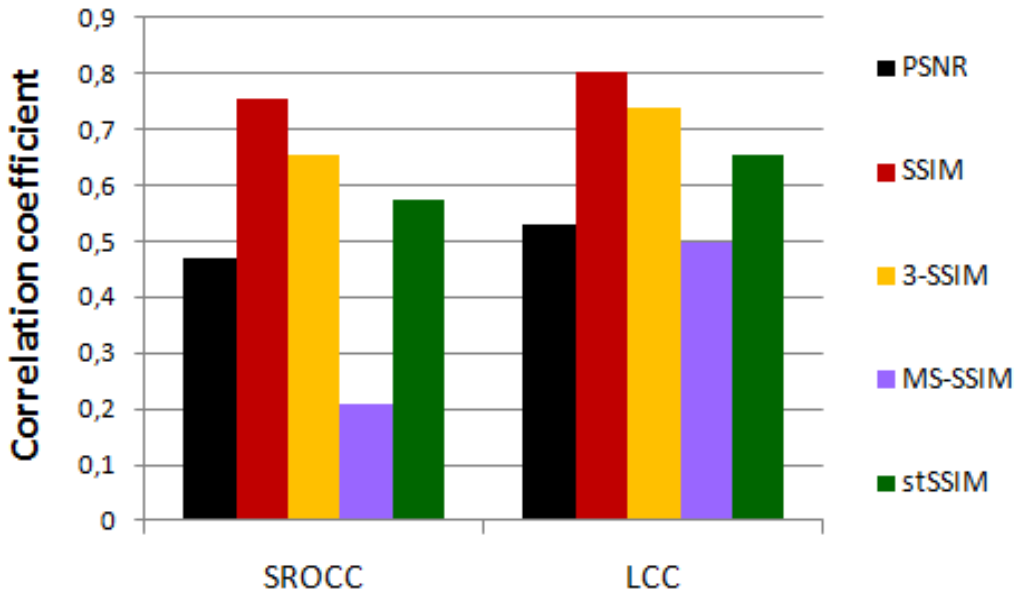
Pearson linear correlation coefficient is a measure of the correlation between two variables X and Y, giving a value between +1 and −1 inclusive. It represents linear dependence between two variables and computes as:

$$r = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}}.$$

In our methodology we perform non-linear regression over logistic function , variables X and Y as described in Video Quality Experts Group documents (http://www.its.bldrdoc.gov/vqeg/projects/projects.php). More information about LCC you can find on the Wikipedia page http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.



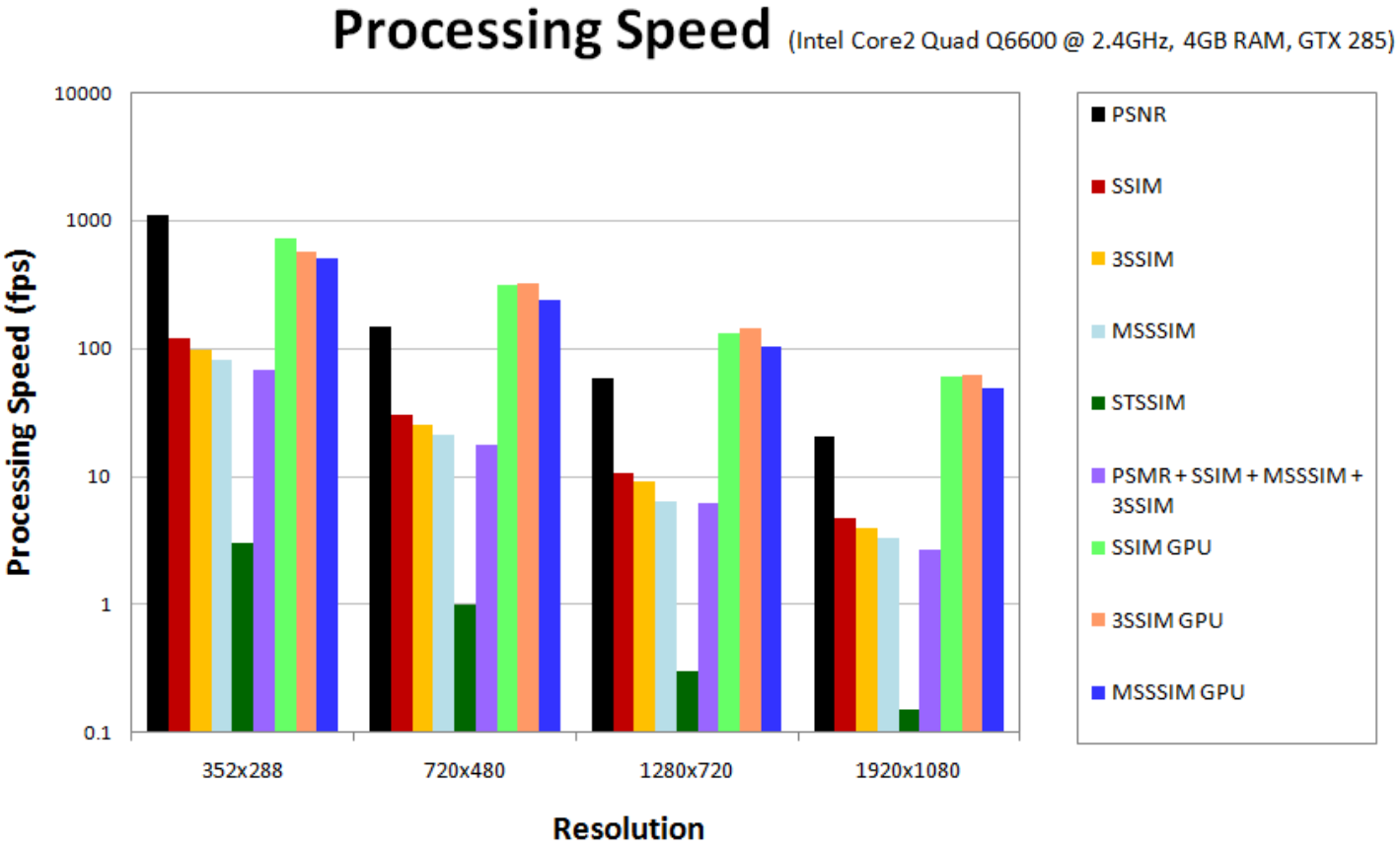
Picture 54. Correlation coefficients for LIVE video quality database



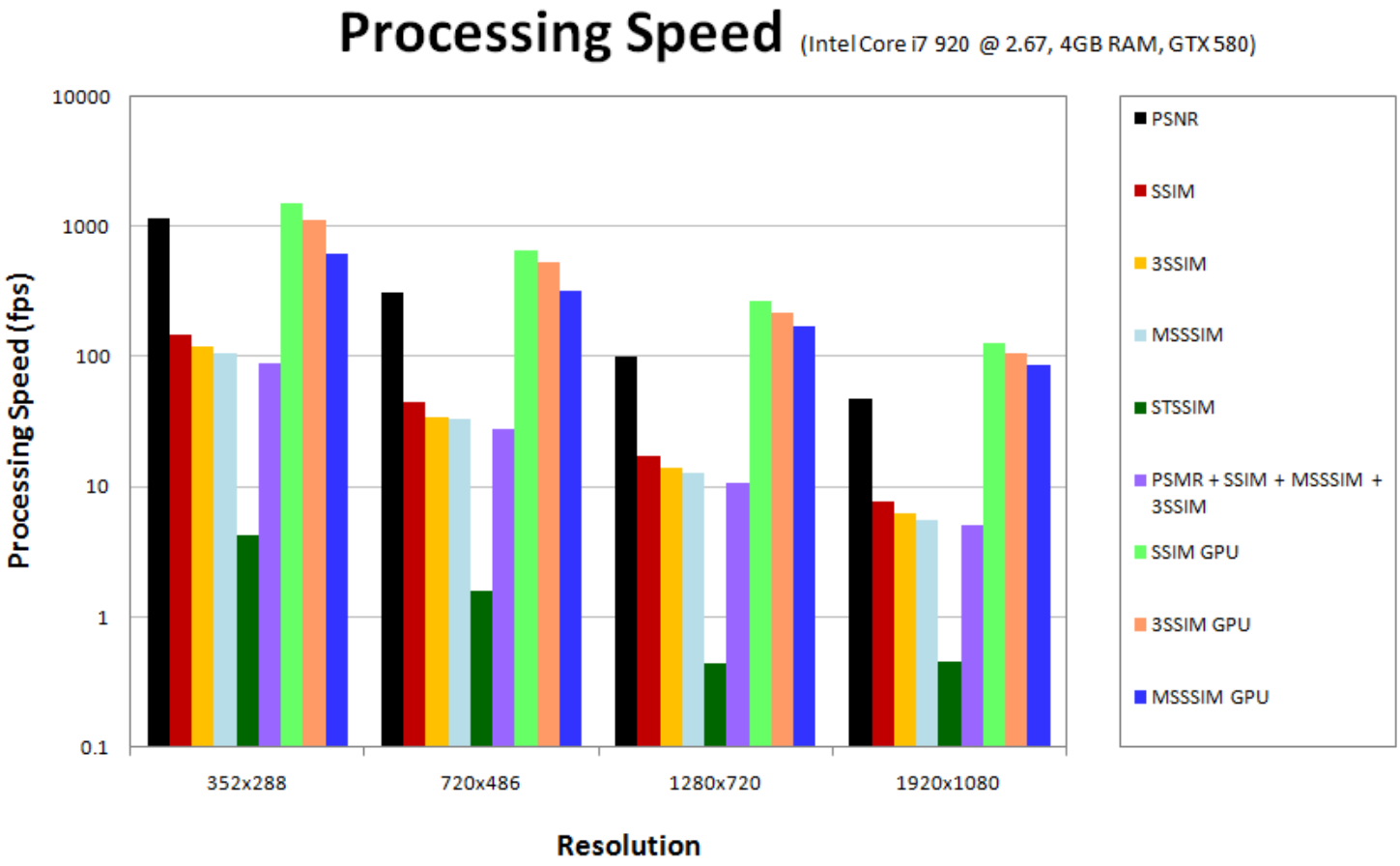
Picture 55. Correlation coefficients for VQEG Phase-I video quality database

Metric speed performance

Measurements of large files can take a very long time. We are always trying to maximize speed of metric implementations using features like multi-threading, SSE\MMX optimizations, high-performance libraries, GPU acceleration. In example, using console interface it is able compute four most popular metrics (PSNR, SSIM, 3-SSIM, MS-SSIM) in almost the same time as the slowest of them. Here we provide metric performance graph for different resolutions for two PC configurations:



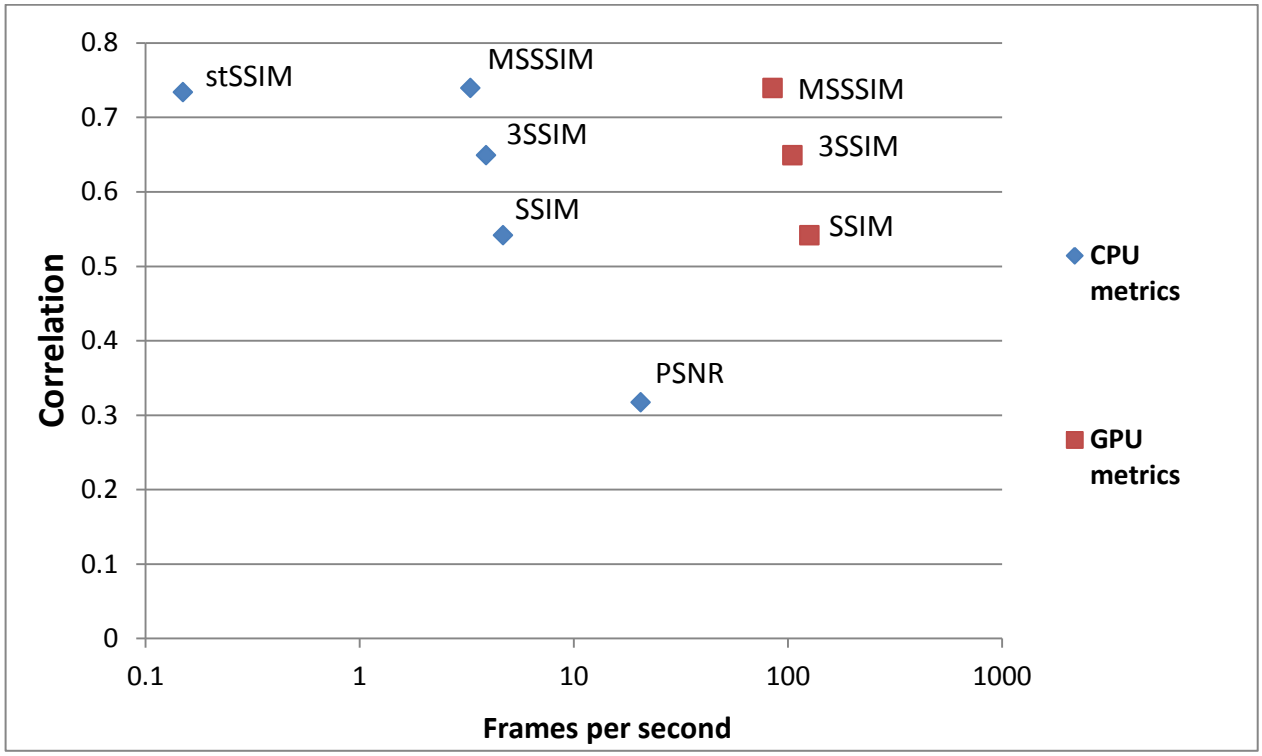
Picture 56. Metrics speed



Picture 57. Metrics speed

Metric speed performance with correlation

We are also providing Speed/Correlation plot, which is allows user to understand difference between metrics. Data provided for 1080p resolution and following configuration: Intel Core i7 920 @ 2.67 GHz, 12GB RAM, NVIDIA GTX 580.



Picture 58. Metrics speed/correlation plot for 1080p resolution

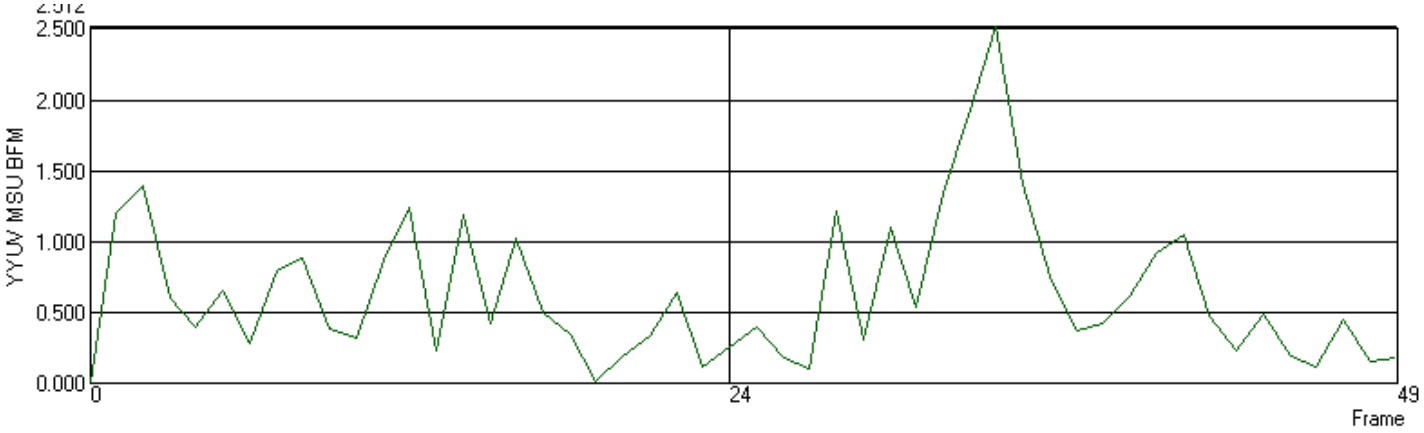
Additional metrics with source code

MSU Brightness Flicking Metric

This metric is made to measure flicking quantity between neighboring frames of the sequence.

Metric's value is modulus of difference between average brightness values of previous and current frames. Resulting metric's value is average value among all per-frame values.

Here are a per-frame metric's values visualized with GUI version of MSU VQMT.



Picture 59. MSU Brightness Flicking Metric, per-frame visualization

MSU Brightness Independent PSNR (BI-PSNR)

This metric is intended for measuring distortions in video taking into account brightness shifts.

Brightness Independent PSNR metric should be used when one of the sequences has any brightness transformation, which does not change within frame. Example of such transformation is uniform increasing of brightness of contrast for single frame of for all sequence. Such transformations prevent usage of standard metrics because of strong brightness difference between comparing frames. BI-PSNR algorithm calculates brightness transformation, which makes frames similar as possible and calculates standard PSNR and MSE metrics taking into account founded transformation.

Algorithm of this metric is next - table C[i,j] is filling for each frame: C[i,j] = { number of points in the same position, which have brighness "i" at the first sequence frame and "j" at the second sequence frame }

Next, for each "i" (brightness value from the first frame) we find corresponding brightness from the second sequence. Following formula is used to estimate distanse from arbitrary values of "i" and "j":

$$W[i,j] = \sum_{k=0}^{255} C[i,k] * (j-k)^2$$

One can note that this formula is sum of quadratic differences between all pixels of the first sequence with value "i" and all corresponding pixels from second sequence on the assumption that brightness was shifted to "i-j".

$$J_i : W[i,J_i] = \min_j(W[i,j])$$

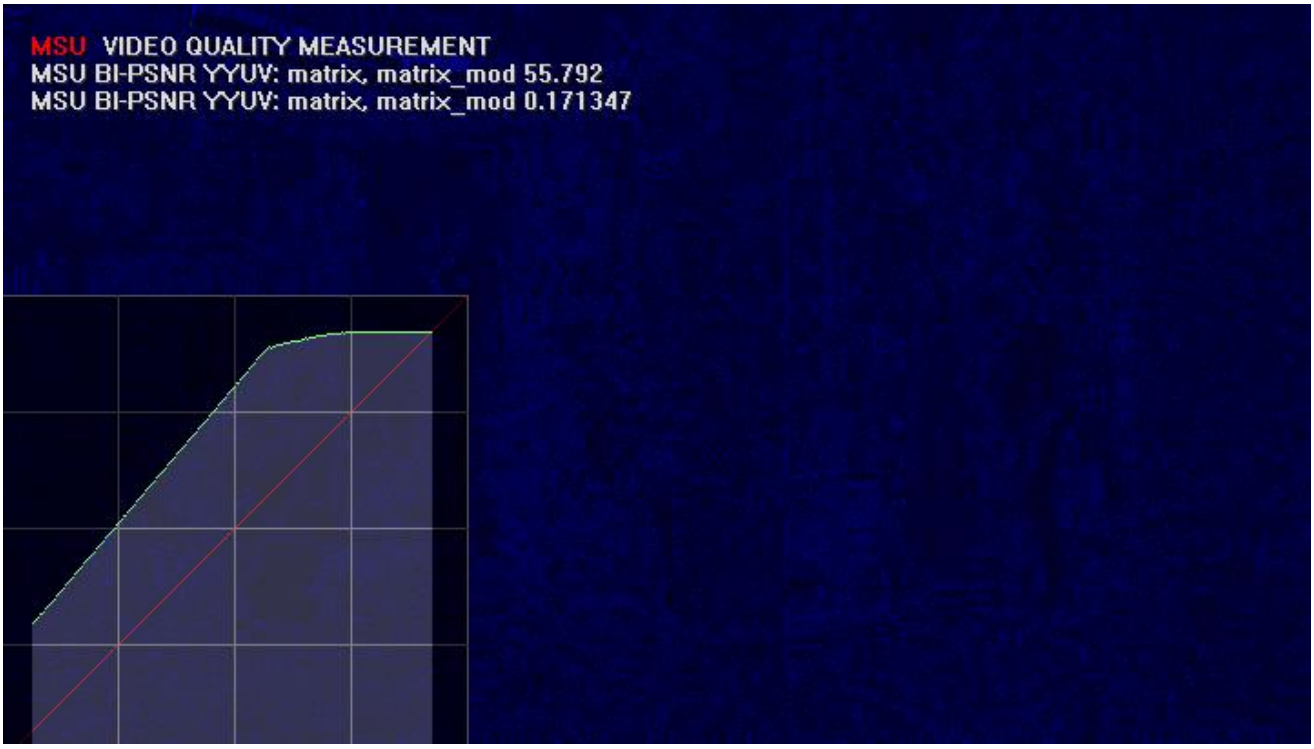
When transformation was found, we can find MSE for the frame taking into account this transformation:

$$MSE = \frac{\sum_i W[i,J_i]}{width * height}$$

There are two part of visualization:

- MSE visualization for frame. Colors of visualization are standard for MSE (in order of error increasing): black-blue-green-red.
- Brightness transformation plot. X-axis is brightness at the first sequence, Y-axis - brightness on the second one. Green points are values, which corresponds to each other (brightness transformation). Red diagonal is identical transformation (no brightness changes).

Here are examples of visualization:



MSU Brightness Independent PSNR visualization

Visualization of the same frame using standard PSNR

Picture 60. MSU Brightness Independent PSNR visualization

MSU Drop Frame Metric

This metric is made to calculate number of drop-frames in a sequence.

Metric's visualization difference of Y-planes between two consecutive frames + 128. So, grey color (128 128 128) means that brightness of a pixel is the same as at the previous frame.

Here is an example of visualization:

Picture 61. MSU Drop Frame Metric visualization

For each frame difference with the previous one is calculating. Metric's value is 1 if frames are identical, 0 otherwise. The resulting metric's value is number of drop-frames.

MSU Noise Estimation metric

This metric is intended for calculation of noise level for each frame of video sequence.

The metrics implements three various algorithms of definition of noise level:

- Frequency domain
- MAD
- Block-Based
- Spatio-Temporal Gradients

The choice the algorithm to use can be made in

- Settings for MSU VQMT GUI version and
- using option –set noise-definition=<value> for MSU VQMT Pro-version, where value can be:
    - frequency_domain
    - MAD
    - block_based
    - spatio_temporal

Frequency domain (since VQMT 11)

This method calculates comparative amount of high frequencies in the 2d-fourier transform of input frame. This method has range (very blurred) 0..1 (very noisy).

See Kanjar De V. Masilamani Image Sharpness Measure for Blurred Images in Frequency Domain. For additional info.

MAD

For each frame do Haar wavelet decomposition. Than evaluate median of HH-component's absolute values. Final value of the metrics is the normalized median.

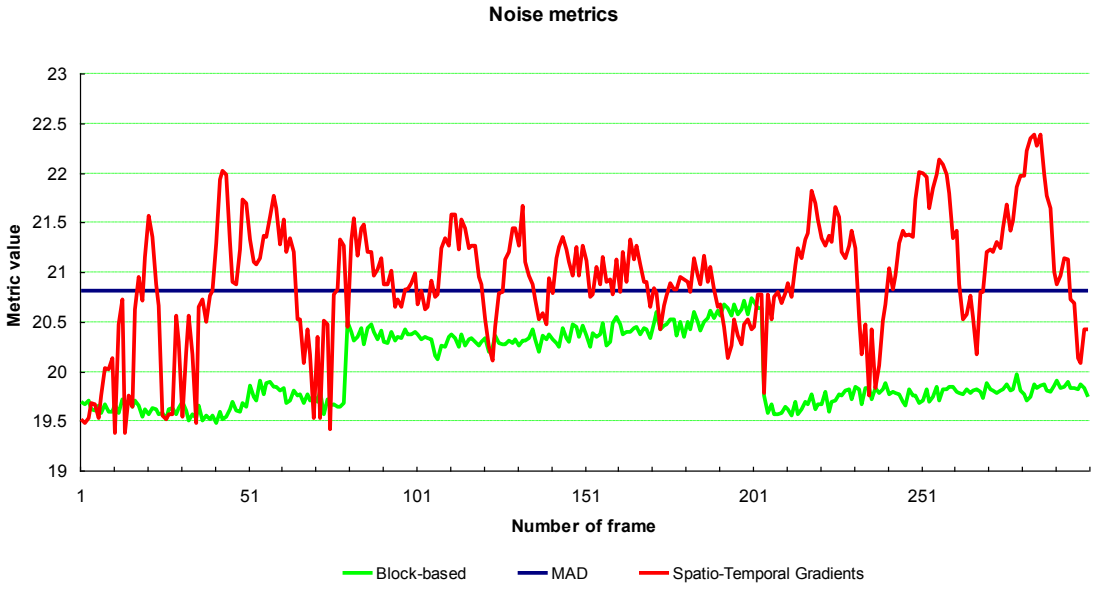
Block-Based

Frames are tessellated into a number of 8x8 blocks. Standard deviations of intensity (metrics of intensity variation) are computed for all the blocks and then sorted. The block with the smallest standard deviation has the least change of intensity. The smaller the standard deviation, the smother the block. The intensity variation of a smooth block may be due to noise, in which the standard deviation of the block is close to that of the Gaussian noise added. Normalized average arithmetic values of 30 % of all blocks with the least values grows is the final value of the metric.

Spatio-Temporal Gradients

Wavelet decomposition for each frame is perform following after temporal and spatial histograms calculation. The initial estimation of noise level is defined by the value at which temporal or spatial histogram achieves the maximal value. The decision of whether use spatial or temporal histogram is based on the deviation of the histogram from the Rayleigh distribution. Then this estimation is corrected using Kolmogorov-Smirnov test. The normalized corrected estimation is the final value of the metric.

Notion estimation visualization (with Excel) example:



Picture 62. MSU Noise Estimation Metrics (Block-based, STG and MAD) visualization

Result of metrics is per-frame value. Final value of the metrics is the arithmetic mean of all per-frame values.

MSU Scene Change Detector

Scene Change Detector is made to automatic identification of scene boundaries in video sequence.

Usage

The plugin implements four algorithms of similarity measurements between two adjacency frames in video sequence:

1. Pixel-level frames comparison
2. Global Histogram comparison
3. Block-Based Histogram comparison
4. Motion-Based similarity metric

The choice of the algorithm can be made in Settings. Numbers from 1 up to 4 corresponds to each algorithm.

Default and recommended value is 3 (Block-Based Histogram).

Visualization

Y-plane is drawing during the visualization. Brightness of scene boundary frames is increased.

Example of visualization:

frame i - 1 frame i frame i + 1

Picture 63. MSU Scene Change Visualization

Plots

Metric's plot is making after all measurements. "One" value means that current frame is the first frame in scene, other frames have "zero" values.

Plot's example:

Picture 64. MSU Scene Change Detector plot

Algorithm

- **Pixel-level comparison.** Similarity metric of two frames is the sum of absolute differences (SAD) between corresponding pixels values.
- **Global Histogram.** The histogram is obtained by counting the number of pixels in frame with specified brightness level. The difference between two histograms is then determined calculating SAD of number of pixels on each brightness level.
- **Block-Based Histogram.** Each frame is divided into 16x16 pixel blocks. Brightness distribution histogram is constructed for each block. Then similarity metric for each block is obtained. Average value of these metrics is accepted as a frames similarity metric.
- **Motion-Based.** Motion Estimation algorithm with block size 16x16 pixels is performed for two adjacency frames at the first stage. After that average value of motion vector errors is accepted as a finally similarity metric.

FAQ

How can I perform batch processing with MSU VQMT?

Use documentation or GUI tool to build command-line, than you can invoke VQMT from script.

Your program reports that it failed to open input file.

We are increasing number of supported formats all the time. If you faced with such problem, please, mail us with attachment of sample file.

What's about values of the metrics? What values for each metrics mean, that quality is better?

Metric	Full Reference (FR)/ No Reference (NR)	Interpretation
PSNR, PSNR (256), APSNR, APSNR (256)	FR	100 for equal frames, higher values are better
SSIM (fast), SSIM (precise), MS-SSIM (fast), MS-SSIM	FR	Higher values are better, 1 for equal frames

(precise), 3SSIM, stSSIM		
VQM	FR	0 for equal frames, lower values are better
MSU Blocking metric	NR	Lower value corresponds to lower blocking.
MSU Blurring metric	NR (but 2 files are necessary in case of measure visualization)	Lower value corresponds to higher blurring
Delta	FR	0 means equal frames, positive and negative values mean deviation, lower absolute values are better
MSAD	FR	0 means equal frames, lower values are better
MSE	FR	Lower values are better, 0 for equal frames
MSU Brightness Flicking Metric	NR	The higher the value, the higher the brightness flicking in comparison with the previous frame
MSU Brightness Independent PSNR	FR	100 for equal frames, higher values are better
MSU Drop Frames Metric	NR	Metric has two values: 0 means that current frame exists, 1 that it is dropped
MSU Noise Estimation Metric	NR	The higher the value, the higher the noise level
MSU Scene Change Detector	NR	Metric has two values: 1 means that current frame is the first frame in a scene, for other frames metric value is 0.

Where can I get more information about your own metrics (MSU Blocking/Blurring metrics and others)?

We plan to prepare some paper first and next publish these assessment methods on our webpage.

How can I add my own metric to your program?

MSU VQMT supports plugins. Please refer to online SDK (https://github.com/msu-video-group/vqmt-sdk) for more information. Also SDK is provided with the installer and is copied into the installation folder.

I receive unequal results for AviSynth scripts. What's wrong?

Try to review resulting video from AviSynth script processing in VirtualDub to find out possible errors in the script.

Why results are unequal (too low) for one or more codecs?

Some codecs (for example DivX 6.0) shift video file one or two frames back, or make two first frames equal. Check for this shift in your video files after encoding. You can open two encoded videos in two different VirtualDub's and switch to the 10th frame, for example. And then do switching between them to see if 10th frames in two videos correspond to each other or not. If shift really exists, use AviSynth scripts with DeleteFrame() and DuplicateFrame() to fix it.

What about masking? I really need it!

Since 2.01 beta MSU VQMT supports masking. Mask will be converted into monochromic image using the following rule: a dark areas of mask files (with all color components less than 128 in RGB case and Y component less than 0 in YUV case) are treated as black, other areas are treated as white.

User can specify mask color (black or white). Masked areas are filled with the specified mask color. User has two choices for mask color definition.

- Specify that the black color is a mask – it means that a black color is taken as a mask color
- Specify that not black color is a mask – it means that a non black color is taken as a mask color

NOTES:

When black color is used as a mask an unmasked area may contain different colors – it is useful when the user wants to mark mask area on an actual video frame

Your masking is not convenient; I want to draw a mask on a video frame!

Actually you can. Specify black color as a mask and draw black mask on frames, everything will be fine.

Why MSU VQMT has four variants of PSNR calculation?

PSNR formula is:

$$PSNR = 10 \cdot \log_{10} \frac{MaxErr^2 \cdot w \cdot h}{\sum_{i=0,j=0}^{w,h} (x_{i,j} - y_{i,j})^2}$$

"PSNR" and "APSNR" use the correct way of PSNR calculation and take maximum possible absolute value of color difference as *MaxErr*. But this way of calculation gives an unpleasant effect after color depth conversion. If color depth is simply increased from 8 to 16 bits, the PSNR will change, because *MaxErr* should change according to maximum possible absolute value of color difference (255 for 8 bit components and 255 + 255/256 for 16 bit components). Thus "PSNR (256)" and "APSNR (256)" are implemented. They would not change because they use upper boundary of color difference as *MaxErr*. The upper boundary is 256. This approach is less correct but it is used often because it is fast.

The difference between "PSNR" and "APSNR" is the same as between "PSNR (256)" and "APSNR (256)" and is in the way of average PSNR calculation for a sequence. The correct way to calculate average PSNR for a

sequence is to calculate average MSE for all frames (average MSE is arithmetic mean of the MSE values for frames) and after that to calculate PSNR using ordinary equation for PSNR:

$$PSNR = 10 * \log_{10} \frac{MaxErr^2}{MSE}$$

This way of average PSNR calculation is used in “PSNR” and “PSNR (256)”. But sometimes it is needed to take simple average of all the per frame PSNR values. “APSNR” and “APSNR (256)” are implemented for this case and calculate average PSNR by simply averaging per frame PSNR values.

The next table summarizes the differences:

Metric	*MaxErr* calculation	Average PSNR calculation
PSNR	correct	correct
PSNR (256)	256 (fast, inexact)	correct
APSNR	correct	averaging
APSNR (256)	256 (fast, inexact)	averaging

“PSNR” metric is recommended for PSNR calculation since it is implemented according to the original PSNR definition.

What about OPSNR? Why MSU VQMT does not have it?

Actually, MSU VQMT does have OPSNR. OPSNR is a shortening from the “Overall PSNR” which means that average PSNR for the sequence is calculated in the “correct” way. Thus “PSNR” and “PSNR (256)” of the MSU VQMT are two implementations of OPSNR. Please refer to “Your masking is not convenient; I want to draw a mask on a video frame!

Actually you can. Specify black color as a mask and draw black mask on frames, everything will be fine.

Why MSU VQMT has four variants of PSNR calculation?” to find out the difference between these two.

Why PSNR is slightly different from previous version?

New version of MSU VQMT uses IPP L2 measure to calculate MSE for PSNR, since IPP does not have direct MSE calculation function. This is a cause of slight difference.

Why MSU VQMT has two variants of SSIM calculation?

SSIM (fast) is identical to the previous version of MSU VQMT SSIM. SSIM (precise) is implemented as a more precise and correct SSIM. It uses Gaussian blur instead of box filter, as SSIM (fast) does.

Why neither of your SSIM calculations does match AviSynth SSIM plugin?

SSIM (fast) is close to AviSynth SSIM plugin. Nevertheless MSU VQMT does not use luminance masking, because it is not used in the original paper (http://ieeexplore.ieee.org/iel5/83/28667/01284395.pdf).

Why SSIM (fast) visualization seems to be shifted?

This effect is caused by the sum calculation algorithm for the box filter. The sum is calculated over the block to the bottom-left or up-left of the pixel (depending on if the image is bottom-up or top-down).

Why MSU Noise Estimation Metric plugin does not match to its previous version?

The previous version of MSU Noise Estimation Metric plugin has a bug: it calculates measure correctly only when MSU VQMT opens file in RGB color space. The bug is fixed now, but color space conversion sequence has changed. The closest result is achieved when conversion algorithm is set to PC.601. But result is not the same because previous conversions were made using integer data, and now the conversion is made using floating point data.

Why MSU Noise Estimation Metric plugin does not match to VirtualDub MSU Noise Estimation filter?

The reason is MSU VQMT and VirtualDub sequences of color space conversions are different. For example, if the file is opened in YV12 color space then MSU VQMT plugin get Y component of the image right away. But VirtualDub plugin gets RGB, converted from YV12 by VirtualDub, and extracts Y component from this RGB. The closest result is achieved when conversion algorithm is set to PC.601. But result is not the same because VirtualDub conversions are made using integer data, and MSU VQMT conversions are made using floating point data.

How can I obtain version for Linux?

Now you can use command line (PRO only) on Linux. For GUI tool please use Wine.

List of figures

Appendix A. Color Spaces Conversion

Overview

The input media for objective quality metrics measurement could be in YUV color space or in RGB color space. MSU VQMT can compute objective metrics in any color space, such as YUV, RGB or L*U*V. And if input content is not in the color space that user wants to use for measurement, color space conversion is performed.

RGB ↔ YUV conversion

The YUV model defines a color space in terms of one luma and two chrominance components. The YUV color model is used in the PAL, NTSC, and SECAM composite color video standards. Previous black-and-white systems used only luma (Y) information and color information (U and V) was added so that a black-and-white receiver would still be able to display a color picture as a normal black and white pictures.

YUV models human perception of color in a different way than the standard RGB model used in computer graphics hardware.

Y stands for the luma component (the brightness) and U and V are the chrominance (color) components.

YUV signals are created from an original RGB (red, green and blue) source. The weighted values of R, G, and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot. The U signal is then created by subtracting the Y from the blue signal of the original RGB, and then scaling; V is created by subtracting the Y from the red, and then scaling by a different factor.

Conversion can be defined by 3×3 matrices C and C' and vectors A and A', such that:

$[R, G, B] = C \times [Y, U, V]^t + A$

$[Y, U, V] = C' \times [R, G, B]^t + A'$,

or by *Kr, Kb, Kg* coefficients, such that

$Y = Kr \cdot R + Kg \cdot G + Kb \cdot B$

$U = \frac{B - Y}{1 - Kb}$

$V = \frac{R - Y}{1 - Kr}$

and then normalization. YUV to RGB is defined as inverse transform.

REC601 table

RGB range: 0..255 0..255 0..255

YUV range: 16..235 16..235 16..235

$$C = \begin{pmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.391 & -0.813 \\ 1.164 & 2.017 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} -222.912 \\ 135.488 \\ -276.928 \end{pmatrix}$$

$$C' = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix}, \quad A' = \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix}$$

PC601 table

RGB range: 0..255 0..255 0..255

YUV range: 0..255 0..255 0..255

$$C = \begin{pmatrix} 1 & 0 & 1.139 \\ 1 & -0.395 & -0.580 \\ 1 & 2.032 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} -145.899 \\ 124.832 \\ -260.110 \end{pmatrix}$$

$$C' = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.1 \end{pmatrix}, \quad A' = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

REC709 table

RGB range: 0..255 0..255 0..255

YUV range: 16..235 16..235 16..235

Kr = 0.2125, Kg = 0.7154, Kb = 0.0721

REC709_c_short table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 0..255 0..255 0..255

Kr = 0.2126, Kg = 0.587, Kb = 0.114

REC709_c_full table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 16..235 16..235 16..235

Kr = 0.2126, Kg = 0.587, Kb = 0.114

REC601_c_short table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 0..255 0..255 0..255

Kr = 0.299, Kg = 0.587, Kb = 0.114

REC601_c_full table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 16..235 16..235 16..235

Kr = 0.299, Kg = 0.587, Kb = 0.114

FCC_c_short table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 0..255 0..255 0..255

Kr = 0.3, Kg = 0.56, Kb = 0.11

FCC_c_full table (new in VQMT 11)

RGB range: 0..255 0..255 0..255

YUV range: 16..235 16..235 16..235

Kr = 0.3, Kg = 0.56, Kb = 0.11

L*U*V* ↔ YUV conversion

YUV to L*U*V*

Y = Y / 255

if (Y > 0.008856)

L = 116 * $Y^{0.33333}$ - 16

else

L = (903.3 * Y)

U and V from L*U*V* color space are equal to U and V in YUV color space.

Appendix B. Structure of a CSV file

CSV file

Each measurement corresponds to one column in result CSV file. First line, that starts with **Metric** describes metric, second line, that starts with **Color** describes color component. Next two lines that start with **File** contain names of the compared files. For metrics, which use only one file for comparison, there can be one line with files. Next several lines contain accumulated value of the metrics for whole sequence. Each of these lines starts with value description (mean, std dev, etc). After these all lines each line contains value of the metric for correspondent frames and starts with the frame number. You can distinguish these lines from accumulated values, because accumulated values first element is never numeric.

Example of a CSV file

Metric	PSNR	NIQE	NIQE
Color	RGB	Y	Y
File	c:\video\loseless.avi	c:\video\loseless.avi	c:\video\cinepack.avi
File	c:\video\cinepack.avi	c:\video\loseless.avi	c:\video\cinepack.avi
total psnr	37.1233		
mean	37.16434	7.369142	7.835093
harmonic mean	37.15495	7.354019	7.805778
min. val	35.79426	6.727007	7.025071
max. val	38.20639	7.819283	9.068192
min. frame	0	1	4
max. frame	1	6	6
std dev	0.587188	0.330279	0.486838
variance	0.344789	0.109084	0.237012
NIQE mean		7.369142	7.835093
0	35.79426	7.022005	7.193083
1	38.20639	6.727007	8.157766
2	36.7478	7.722941	8.092119
3	37.35955	7.401671	8.058648
4	36.97003	7.099359	7.025071
5	37.33134	7.25418	8.360405
6	37.47937	7.819283	9.068192
7	37.01985	7.371838	8.263447
8	37.31721	7.643652	7.679767
9	37.41765	7.629482	7.952714
10			7.653012
11			7.335667
12			7.49974
13			7.601908
14			7.76027

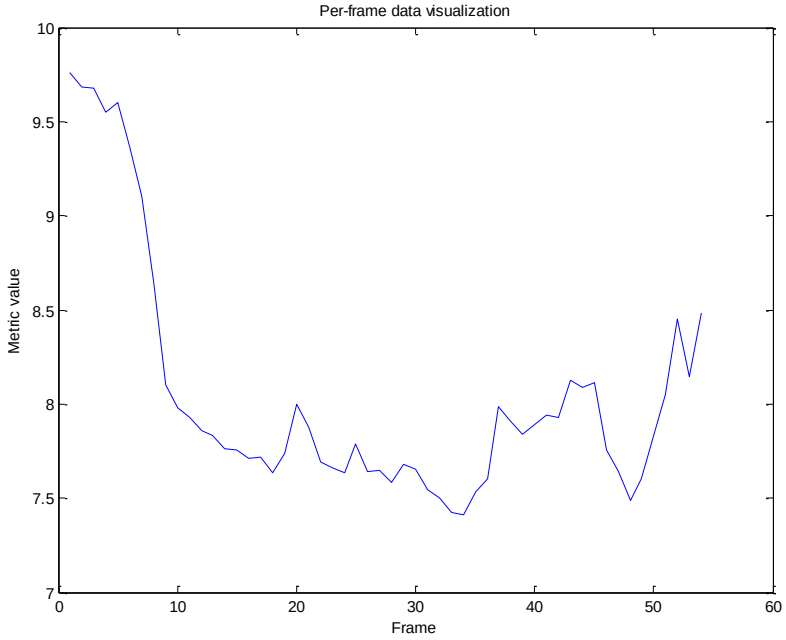
15			7.65969

Parsing in MATLAB

MATLAB internal function *importdata()* properly works with generated CSV files. Script below demonstrates parsing and visualization of per-frame CSV files:

```
A = importdata('per_frame_res.csv');
plot(A.data(:, 1));
xlabel('Frame');
ylabel('Metric value');
title('Per-frame data visualization');
```

Picture 65 show results of this script execution.

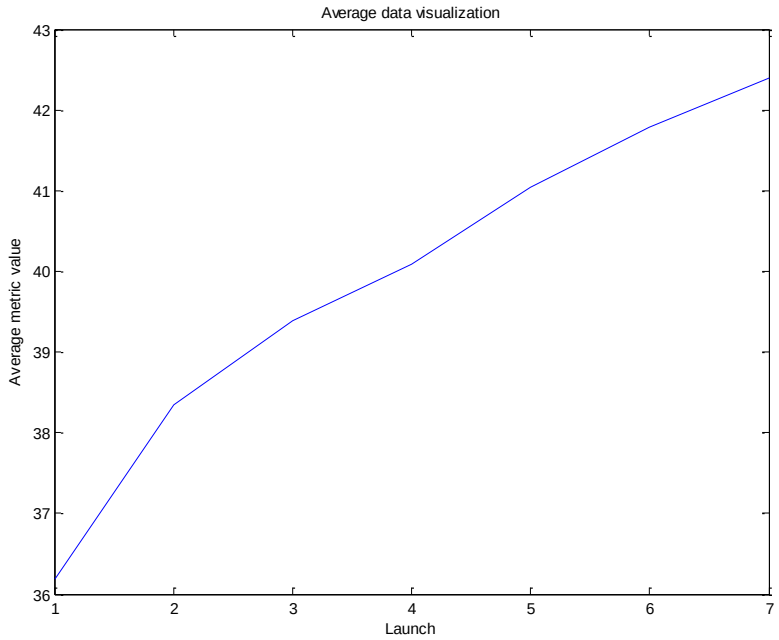


Picture 65. Example of per-frame results visualization using MATLAB

Parsing of the average results file can be performed with the same way:

```
A = importdata('avg_res.csv');
plot(A.data(:, 1));
xlabel('Launch');
ylabel('Average metric value');
title('Average data visualization');
```

Picture 66 show results of this script execution.



Picture 66. Example of per-frame results visualization using MATLAB

About us (Graphics & Media Lab Video Group)

Graphics&Media Lab Video Group is a part of Graphics&Media Lab of Computer Science Department in Moscow State University. The history of Graphics Group began at the end of 1980's. Graphics&Media Lab was officially founded in 1998. Main research directions of the lab lie in different areas of Computer Graphics, Computer Vision and Media Processing (audio, image and video processing). Some of research results were patented, other results were presented in a number of publications.

Main research directions of Graphics&Media Lab Video Group are video processing (pre-, post- and video analysis filters) and video compression (codecs' testing and tuning, quality metrics research, development of codecs).

Our main achievements in video processing:

- High quality industrial filters for format conversion including high quality deinterlacing, high quality frame rate conversion, new fast practical super resolution, etc.
- Methods for modern TV-sets: big family of up-sampling methods, smart brightness and contrast control, smart sharpening, etc.
- Artifacts' removal methods: family of denoising methods, flicking removal, video stabilization with frame edges restoration, scratches, spots, drop-outs removal, etc.
- Specific methods like: subtitles removal, construction of panorama image from video, video to high quality photo, video watermarking, video segmentation, practical fast video deblur, etc.

Our main achievements in video compression:

- Well-known public comparisons of JPEG, JPEG-2000, MPEG-2 decoders, MPEG-4 and annual H.264 codec's testing; also we provide tests for "weak and strong points of codec X" for companies with bugreports and codec tuning recommendations.
- Our own video quality metrics research, public part is MSU Video Quality Measurement Tool and MSU Perceptual Video Quality Tool.
- We have internal research and contracts on modern video compression and publish our MSU Lossless Video Codec and MSU Screen Capture Video Codec – codecs with ones of the highest compression ratios.

We are really glad to work many years with companies like Intel, Samsung, RealNetworks and others.

A mutual collaboration in areas of video processing and video compression is always interesting for us.

E-mail: video@graphics.cs.msu.ru