### Contents

Overview ........................................................................................................................................5
Brief Description ..............................................................................................................................5
Change Log.....................................................................................................................................6
Main features ..................................................................................................................................10

Command Line Options .................................................................................................................12

Batch files examples .......................................................................................................................17

- Compare two AVI files using PSNR metric for Y component of YUV color space, save CSV file "results.csv" to "D:\videotest\csv" ........................................................................................................17
- Compare two AVI files with original, save CSV........................................................................17
- Compare original YUV file and processed AVI file using PSNR metric for Y component of YUV color space, save CSV file with results to "D:\videotest\csv"................................................18
- Compare original YUV file and processed AVI file using PSNR metric for Y component of YUV color space plus a mask file with a black area as a mask, save CSV file with results to "D:\videotest\csv"................................................18
- Compare original AVI file and compressed MPG file using PSNR metric for Y component of YUV color space, save CSV file with results to "D:\videotest\csv" (MPG file is opened through D:\videotest\compressed.avs) ..................................................................................................................19
- Compute all possible metrics for two AVI files, save CSV file with results to "D:\videotest" .....19
- Compute MSE and PSNR metrics on Y components of YUV color space for two AVI files, save file "avg.csv" with average values and csv file with per frame values into the current folder ....20
- Compute PSNR of all supported color components for two AVI files, save file "avg.csv" with average values and cv file with per frame values into the current folder ........................................................................20
- Compute PSNR on Y component of YUV color space and save visualization video for two AVI files in current folder ..................................................................................................................21
- Compare all AVI files from folder1 with file folder2\input.avi with all objective quality metrics ...21
- Compute blocking and blurring metrics for all AVI files in current folder in its subfolders ....22

Information about metrics .............................................................................................................23
Overview ........................................................................................................................................23

Delta ...............................................................................................................................................24

- Brief Description ..........................................................................................................................24
- Examples ......................................................................................................................................24

MSE ................................................................................................................................................27

- Brief description ..............................................................................................................................27
- Examples ......................................................................................................................................27

MSAD ...............................................................................................................................................30

- Examples ......................................................................................................................................30

PSNR (Peak Signal-to-Noise Ratio) ..................................................................................................33

- Brief Description ..............................................................................................................................33
- Examples ......................................................................................................................................34

SSIM INDEX .......................................................................................................................................39

- Brief Description ..............................................................................................................................39
- Examples ......................................................................................................................................41

MultiScale SSIM INDEX ...................................................................................................................46

- Brief Description ..............................................................................................................................46
- Examples ......................................................................................................................................49

---

FAQ

How can I perform job processing with MSU VQMT? ................................................................. 95
Your program reports that it failed to open input file................................................................. 95
What's about values of the metrics? What values for each metrics mean, that quality is better? ............................................................... 96
Where can I get more information about your own metrics (MSU Blocking/Blurring metrics
and others)? ........................................................................................................................ 96
How can I add my own metric to your program? ................................................................. 97
I receive unequal results for AviSynth scripts. What's wrong? ............................................. 97
Is it possible, to open video files, which are not supported by MSU VQMT directly (.mp4,
.mpg, .mov etc.)? ............................................................................................................... 97
Why results are unequal (too low) for one or more codecs? ................................................... 97
What about masking? I really need it! ...................................................................................... 97
Your masking is not comfortable; I want to draw a mask on a video frame! ................................. 98
Why MSU VQMT has four variants of PSNR calculation? ....................................................... 98
What about OPSNR? Why MSU VQMT does not have it? ..................................................... 98
Why PSNR is slightly different from previous version? .............................................................. 98
Why MSU VQMT has two variants of SSIM calculation? ......................................................... 99
Why neither of your SSIM calculations does match AviSynth SSIM plugin? ............................. 99
Why SSIM (fast) visualization seems to be shifted? ................................................................. 99
Why MSU Noise Estimation Metric plugin does not match to its previous version? .............. 99
Why MSU Noise Estimation Metric plugin does not match to VirtualDub MSU Noise
Estimation filter? ............................................................................................................. 99
How can I obtain version for Linux? .......................................................................................... 99
List of figures .................................................................................................................. 100

Appendix A. Color Spaces Conversion ........................................................................... 101
  Overview .................................................................................................................... 101
  RGB <> YUV conversion ............................................................................................ 101
    REC.601 Table ....................................................................................................... 101
    PC.601 Table ......................................................................................................... 101
  L"U"V" <> YUV conversion ........................................................................................ 102
    YUV to L"U"V" ..................................................................................................... 102

Appendix B. Structure of a CSV file ............................................................................. 103
  CSV file ..................................................................................................................... 103
    Structure of CSV file using regular expressions .................................................... 103
    Example of CSV file ............................................................................................. 103
  Average CSV file ...................................................................................................... 104
    Structure of CSV file using regular expressions .................................................... 104
    Example of average CSV file ................................................................................. 104
  Average CSV file, second version ............................................................................ 105
    Structure of CSV file using regular expressions .................................................... 105
    Example of average CSV file ................................................................................. 105
  Parsing in MATLAB ................................................................................................. 106

About us (Graphics & Media Lab Video Group) .......................................................... 108

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 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](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](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](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 Console 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.
Change Log

[!] – Known bug
[+] – New Feature
[*] – Other

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
[+] 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 average 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:**
  - AVI-container (for any installed Video for Windows decoder)
  - Raw files with 8, 10, 14 and 16 bit color depth (YV12, UYVY, YUY2, YUV, IYUV, PXXX color spaces)
  - AviSynth scripts – it can be very useful when using VOB, WMV and other files as input for MSU VQMT
  - MOV, VOB, WMV, MP4, MPG, MKV, FLV and other known video containers via AviSynth scripts auto generation

- **13 base metrics are included:**
  - PSNR
  - PSNR (256)
  - APSNR
  - APSNR (256)
  - MSE
  - MSAD
  - Delta
  - SSIM (fast)
  - SSIM (precise)
  - MSSSIM (fast)
  - MSSSIM (precise)
  - 3SSIM
  - stSSIM
  - VQM

- **7 additional objective quality metrics from MSU:**
  - MSU Blurring
  - MSU Blocking
  - 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)

  - All metrics can be calculated simultaneously
  - Every metric can be calculated for its set of color planes (Y,U,V,L,R,G,B)
  - Masking
  - Results are saved to CSV files, or visualized with GUI
  - 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>`

Parameters are described in the next table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-f &lt;filename&gt;</code></td>
<td>File to calculate metric for. For .yuv files you need to specify color space:</td>
<td>YUV</td>
</tr>
<tr>
<td><code>-f &lt;filename&gt; &lt;color_space&gt;</code></td>
<td>where color_space is one of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YUV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YV12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IYUV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• UYUV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YUY2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P010 (10 bit 4:2:0 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P014 (14 bit 4:2:0 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P016 (16 bit 4:2:0 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P210 (10 bit 4:2:2 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P214 (14 bit 4:2:2 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P216 (16 bit 4:2:2 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P410 (10 bit 4:4:4 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P414 (14 bit 4:4:4 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P416 (16 bit 4:4:4 yuv)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P410_RGB (10 bit 4:4:4 rgb)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P414_RGB (14 bit 4:4:4 rgb)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P416_RGB (16 bit 4:4:4 rgb)</td>
<td></td>
</tr>
</tbody>
</table>

More info 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.
First file is assumed as original.

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

-mask <filename> BLACK | NOT_BLACK

Specification of a mask file. Such file is specified in the same way as described for “-f” 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.

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.

User can specify mask color. Masked areas are filled with the specified mask color. User has two choices for mask color definition.

- **BLACK** – it means that a black color is taken as a mask color
- **NOT_BLACK** – it means that a non black color is taken as a mask color

NOTES:

- we assume the color with three zero components in RGB and with zero Y-component in YUV as “black”
- if, for example, the mask file is provided in RGB and calculations are performed in YUV, then mask color is converted into YUV using provided conversion settings and the masked areas are filled with the color achieved
- 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

-metr <metric_name> -cc <color_component> [-cl configuration_line]

Metric & and color components for calculation. Metric name is one of the

- psnr
- psnr_256
- apsnr
- apsnr_256
- delta
- msad
- mse
- ssim_fast
- ssim_precise
- 3ssim
- msssim_fast
- msssim_precise
- vqm
- blurring_metric
- blocking_metric
- name of the metric from plug-in (written with SDK)
- ALL key to calculate ALL metrics (including plug-ins) with ALL supported color components.

or one of the CUDA metrics (CUDA-capable device required):

- ssim_cuda
- 3ssim_cuda
- msssim_cuda

Color component is one of the

- YYUV
- UYUV
- VYUV
- LLUV
- RRGB
- GRGB
- BRGB.

Specify ALL key to compute metric for all
supported color components; specify YUV key to compute metric for all YUV color components; specify RGB key to compute metric for all RGB color components.

If metric is configurable, use -cl parameter to set configuration string. Please refer to the metric home page to find out applicable configuration line.

```
-supported color components; specify YUV key to compute metric for all YUV color components; specify RGB key to compute metric for all RGB color components.

If metric is configurable, use -cl parameter to set configuration string. Please refer to the metric home page to find out applicable configuration line.

-sc 0 | 1
-ryt REC601|PC601
-cng CUSTOM <filename> | POSTFIX
-cod <dirname>
-sbf 0 | 1
-bfod <dirname>
-bft ORIGPROC | PROCPROC | FIRSTBETTER | SECONDBETTER
-bfnum
-bfr
-lp <filename>
-af <filename>
-af2 <filename>
-yw <width>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-yh &lt;height&gt;</td>
<td>Height of input YUV file.</td>
</tr>
<tr>
<td>-fpd 0</td>
<td>1</td>
</tr>
<tr>
<td>-ct 0</td>
<td>1</td>
</tr>
<tr>
<td>-sv 0</td>
<td>1</td>
</tr>
<tr>
<td>-list</td>
<td>Print information for all metrics and plug-ins.</td>
</tr>
<tr>
<td>-vod &lt;dirname&gt;</td>
<td>Directory, where visualization video files will be saved.</td>
</tr>
</tbody>
</table>
Batch files examples

Here are the examples for most popular tasks, which are performed with MSU VQMT PRO version.

Compare two AVI files using PSNR metric for Y component of YUV color space, save CSV file "results.csv" to "D:\videotest\csv"

```
msu_metric -f "D:\videotest\video1.avi" -f "D:\videotest\video2.avi" -metr psnr -cc YYUV -sc 1 -cng CUSTOM "results.csv" -cod "D:\videotest\csv"
```

Compare two AVI files with original, save CSV

```
msu_metric -f "D:\videotest\original.avi" -f "D:\videotest\processed1.avi" -f "D:\videotest\processed2.avi" -metr psnr -cc YYUV -sc 1 -af2 "avg.csv"
```
Compare original YUV file and processed AVI file using PSNR metric for Y component of YUV color space, save CSV file with results to "D:\videotest\csv"

```
msu_metric.exe -f "D:\videotest\original.yuv" IYUV -yw 352 -yh 288 -f
"D:\videotest\processed.avi" -sc 1 -cod "D:\videotest\csv" -metr psnr -cc YYUV
```

Compare original YUV file and processed AVI file using PSNR metric for Y component of YUV color space plus a mask file with a black area as a mask, save CSV file with results to "D:\videotest\csv"

```
msu_metric.exe -f "D:\videotest\source.yuv" IYUV -yw 352 -yh 288 -f
"D:\videotest\processed.avi" -mask "D:\videotest\mask.avi" BLACK -sc 1 -cod
"D:\videotest\csv" -metr psnr -cc YYUV
```
Compare original AVI file and compressed MPG file using PSNR metric for Y component of YUV color space, save CSV file with results to "D:\videotest\csv\" (MPG file is opened through D:\videotest\compressed.avs)

Compute all possible metrics for two AVI files, save CSV file with results to "D:\videotest"
Compute MSE and PSNR metrics on Y components of YUV color space for two AVI files, save file "avg.csv" with average values and csv file with per frame values into the current folder

\[
\text{msu\_metric -f "D:\videotest\video1.avi" -f "D:\videotest\video2.avi" -metr psnr -cc YYUV -metr mse -cc YYUV -sc 1 -af2 "avg.csv"}
\]

Compute PSNR of all supported color components for two AVI files, save file "avg.csv" with average values and csv file with per frame values into the current folder

\[
\text{msu\_metric -f "D:\videotest\video1.avi" -f "D:\videotest\video2.avi" -metr psnr -cc ALL -sc 1 -af "avg.csv"}
\]
Compute PSNR on Y component of YUV color space and save visualization video for two AVI files in current folder

```
msu_metric -f "D:\videotest\video1.avi" -f "D:\videotest\video2.avi" -metr psnr -cc YYUV -sv 1
```

Compare all AVI files from folder1 with file folder2\input.avi with all objective quality metrics

```
for /R folder1 %F in (*.avi) do msu_metric -f "folder2\input.avi" -f %F -metr ALL -sc 1 -cng CUSTOM "%~nF_results.csv" -af2 "average.csv"
```

**Note:** if you use it in batch file, please use %%F instead %F and %%~nF instead of %~nF
Compute blocking and blurring metrics for all AVI files in current folder in its subfolders

*avi

MSU VQMT command line tool

csv file with Y-blurring and Y-blocking

for /R . %F in (*.avi) do msu_metric -f %F -metr blurring_metric -cc YYUV -metr blocking_metric -cc YYUV -sc 1 -cng CUSTOM "%~nF_results.csv"

Note: if you use it in batch file, please use %~xF instead %F and %~nxF instead of %~nF
Information about metrics

Overview

Current version of MSU VQMT PRO contains 14 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
  - VQM

- **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

- **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.

\[ d(X, Y) = \frac{\sum_{i,j} (X_{i,j} - Y_{i,j})}{mn} \]

The values are in -255..255. 0 – for identical frames.

Examples
Here is example of this metric

![First frame](image1.png) ![Second frame](image2.png) ![Delta](image3.png)

Picture 1. Delta example for two frames

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

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.

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

The values are in 0..65025. 0 – for identical frames.

Examples
Here is example of this metric:

![Original](image1.png) ![Processed](image2.png) ![MSE](image3.png)

Picture 4. MSE example for two frames
Here are more examples how different distortions influence on MSE value.

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.

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

The values are in 0..255. 0 – for identical frames.

Examples

Here is example of this metric visualization:

![Original](image1.png) ![Processed](image2.png) ![MSAD](image3.png)

Picture 7. MSAD example for two frames
Here are more examples how different distortions have influence on MSAD value.

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.

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

where \( \text{MaxErr} \) — 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. PSNR metric is easy and fast to calculate, but sometimes it is not appropriate to human's perception.

In MSU VQMT there are four PSNR implementations. "PSNR" and “APSNR” use the correct way of PSNR calculation and take maximum possible absolute value of color difference as \( \text{MaxErr} \). 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” and "APSNR" will change, because \( \text{MaxErr} \) 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 \( \text{MaxErr} \). The upper boundary is 256. This approach is less correct but it is used often because it is fast. Here are the rules of \( \text{MaxErr} \) definition:

- “PSNR” and “APSNR” (psnr and apsnr in command prompt) – \( \text{MaxErr} \) varies on color components bits usage:
  - 255 for 8 bit components
  - 255 + 3/4 for 10 bit components
  - 255 + 63/64 for 14 bit components
  - 255 + 255/256 for 16 bit components
  - 100 for L component of LUV color space

Notes

- If bits depth differs for two compared videos, then maximum bits usage is taken to select \( \text{MaxErr} \).
  - All color space conversions are assumed to lead to 8 bit images. It means that if, for example, you are measuring R-RGB PSNR for 14 bit YUV file, then 255 will be taken as \( \text{MaxErr} \).

- “PSNR (256)” and “APSNR (256)” (psnr_256 and apsnr_256 in command prompt) - \( \text{MaxErr} \) is selected according to the next rules:
  - 256 for YUV and RGB color spaces
  - 100 for L component of LUV color space

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:

\[
PSNR = 10 \times \log_{10} \frac{MaxErr^2}{MSE}
\]

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:

<table>
<thead>
<tr>
<th>Metric</th>
<th>MaxErr calculation</th>
<th>Average PSNR calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>correct</td>
<td>correct</td>
</tr>
<tr>
<td>PSNR (256)</td>
<td>256 (fast, inexact)</td>
<td>correct</td>
</tr>
<tr>
<td>APSNR</td>
<td>Correct</td>
<td>averaging</td>
</tr>
<tr>
<td>APSNR (256)</td>
<td>256 (fast, inexact)</td>
<td>averaging</td>
</tr>
</tbody>
</table>

All the implemented PSNR metrics values are in 0..100. 100 – for identical frames.

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

Examples

PSNR visualization uses different colors for better visual information:

- Black – Metric value is very small (99..100)
- Blue – metric value is small (35..99)
- Green – metric value is medium (20..35)
- Yellow – metric value is high (17..20)
- Red – metric value is very high (0..17)

Here is example of this metric:
Picture 10. PSNR example for two frames
Here are more examples how different distortions have influence on PSNR value.

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
- MSU JPEG 2000 Image Codecs Comparison
- First Annual MSU H.264/MPEG-4 AVC Video Codec Comparison
- Second Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
- MPEG-2 Video Decoders Comparison
- MSU Windows Media Photo (Microsoft HD Photo) and JPEG 2000 Codec Comparison
- Third Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison
- Lossless Video Codec Comparison ‘2007
- Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison

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

![RD-curve](http://www.compression.ru/video/quality_measure/vqmt_pro_en.html)
SSIM INDEX

Brief Description


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:

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

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:

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

where

---

\[\mu_x = \sum_{i=1}^{N} \omega_i x_i\]
\[\sigma_x = \left(\sum_{i=1}^{N} \omega_i (x_i - \mu_x)^2\right)^{1/2}\]
\[\sigma_{xy} = \sum_{i=1}^{N} \omega_i (x_i - \mu_x)(y_i - \mu_y)\]

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

- \(C_1 = 0.01 \times 0.01 \times \text{video1Max} \times \text{video2Max}\)
- \(C_1 = 0.03 \times 0.03 \times \text{video1Max} \times \text{video2Max}\)

where \(\text{video1Max}\) is the maximum value of a given color component for the first video, \(\text{video2Max}\) 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

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 Gauss blur.

Fast implementation visualization seems to be shifted. This effect originates from 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. Value 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 system, 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.

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 visualization for image with itself, value = 1](image1)

![SSIM (precise) for image with noisy image, value = 0.581786](image2)

![SSIM (precise) for image with blurred image, value = 0.923186](image3)

![SSIM (precise) for image with sharpen image, value = 0.976667](image4)

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

- Third Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison

- Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison

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**

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

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

$$c_i(x, y) = \frac{(2\sigma_x\sigma_y + C_2)}{\sigma_x^2 + \sigma_y^2 + C_2}$$ – 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

\[
\mu_x = \sum_{i=1}^{N} \omega_i x_i \\
\sigma_x = \left( \sum_{i=1}^{N} \omega_i (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \\
\sigma_{xy} = \sum_{i=1}^{N} \omega_i (x_i - \mu_x)(y_i - \mu_y)
\]

and \(\alpha_i, \beta_i, y_i\) – 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:

- \(C1 = 0.01 \times 0.01 \times \text{video1Max} \times \text{video2Max}\)
- \(C2 = 0.03 \times 0.03 \times \text{video1Max} \times \text{video2Max}\)
- \(C3 = 0.5 \times C2\)

where \(\text{video1Max}\) is the maximum value of a given color component for the first video, \(\text{video2Max}\) 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):

- \(\text{videoMax} = 255\) for 8 bit color components
- \(\text{videoMax} = 255 + 3/4\) for 10 bit color components
- \(\text{videoMax} = 255 + 63/64\) for 14 bit color components
- \(\text{videoMax} = 255 + 255/256\) for 16 bit color components

Same as in SSIM, there are 2 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 Gauss blur.

Fast implementation visualization seems to be shifted. This effect originates from 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](image1.jpg) ![Compressed](image2.jpg)
Picture 21. MSSSIM example for compressed video

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

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

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 $p_d(x, y) \leq TH1$, then the pixel is regarded as part of a smooth region.
- otherwise, the pixel is regarded as part of a textured region.

---

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

\[ 3\text{SSIM}(x, y) = \frac{\alpha \sum SSIM(x, y)^{\text{Tex}} + \beta \sum SSIM(x, y)^{\text{Edge}} + \gamma \sum SSIM(x, y)^{\text{Smooth}}}{\text{width} \times \text{height}} \]

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

and \( \alpha, \beta, \gamma \) – 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](image1)

![Image with added noise](image2)

![Blurred image](image3)

![Sharpen image](image4)

*Picture 27. Original and processed video sequences (for SSIM example)*
Here are the regions (same colors as in previous example):

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

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 Austin, “Efficient Motion Weighted Spatio-Temporal Video SSIM Index”, 2010, USA

This article could be found here:

The idea of this algorithm is to use motion-oriented weighted windows for SSIM Index. MSU Motion Estimation algorithm 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 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 = -0.0017226

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:

\[
VQM = \text{meart}(|D|) + 0.005 \cdot \max(|D|)
\]

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](image1) ![Processed](image2) ![VQM](image3)

**Picture 30. VQM example for processed image**

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

![Original image](image4) ![Image with added noise](image5)

![Original image](image6) ![Image with added noise](image7)
MSU VIDEO QUALITY MEASUREMENT TOOL 3.0  CS MSU GRAPHICS & MEDIA LAB
MOSCOW STATE UNIVERSITY, JULY 2011  VIDEO GROUP

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


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

![RD-curve example](Picture_33.png)

**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.

The 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:

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

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):

\[ V_{\text{blurring}} = |A_1 - A_2| + |B_1 - B_2| \]

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.
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

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

![MSU Blurring graph](http://www.compression.ru/video/quality_measure/vqmt_pro_en.html)

**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, it allows to measure blocking more precisely. We use information from previous frames to achieve better accuracy.

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:

\[ V_i = V_{i-1} + V_{i+1} \]

\[ V_{i+1} = V_{i-1} + V_i \]
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](image)

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

$$
W_1 = W(ab|A[0]| + ab|B[0]|)
$$

$$
$$

$$
$$

$$
W_R = (W_1 + W_3) \cdot W_2
$$

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

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

**Examples**

Example of blocking metric visualization is shown at Picture 42.

![Picture 42](A. Decoded frame. B. Visualization of blocking metric)

Here are examples how compression has influence on MSU Blocking value.
And here are the MSU Blurring values of Y-plane for these images

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

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

![MSU Blocking Graph](http://www.compression.ru/video/quality_measure/vqmt_pro_en.html)

**Picture 45.** MSU Blocking example
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 hardware. For implementations CUDA technology was used. SSIM, 3-SSIM, and MS-SSIM metrics implemented for now.

Our CPU Implementation of the metrics above are using Intel IPP. We don't know precision and operations order of these algorithm in the IPP, so metric values for GPU implementations can slightly differ from the CPU.

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:

![SSIM metric speedup](Picture 46)
Picture 47. SSIM metric fps graph

Picture 48. 3SSIM metric speedup

Picture 49. 3SSIM metric fps graph

Picture 50. MSSSIM metric speedup
Picture 51. 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:

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


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:

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

Figure 52. Correlation coefficients for LIVE video quality database

Figure 53. 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:

**Processing Speed** (Intel Core2 Quad Q6600 @ 2.4GHz, 4GB RAM, GTX 285)
Metric speed performance with correlation

We are also providing Speed/Correlation plot, which 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 56. Metrics speed/correlation plot for 1080p resolution

[Diagram showing CPU and GPU metrics for various correlation values and frames per second]
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 57. 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: $C[i,j] = \{\text{ number of points in the same position, which have brightness } "i" \text{ at the first sequence frame and } "j" \text{ at the second sequence frame } \}$

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

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

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:

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

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 58. **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:

![MSU Drop Frame Metric visualization](image)

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:

- 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 --cl <name of metric> for MSU VQMT Pro-version

**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 than sorted. The block with the smallest standard deviation has the least change of intensity. The smaller the standard deviation, the smoother 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:

![Graph showing Noise metrics with Excel visualization](image)

**Picture 60.** 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 61. 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 62. 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 job processing with MSU VQMT?
- Your program reports that it failed to open input file.
- What's about values of the metrics? What values for each metrics mean, that quality is better?
- Where can I get more information about your own metrics (MSU Blocking/Blurring metrics and others)?
- How can I add my own metric to your program?
- I receive unequal results for AviSynth scripts. What's wrong?
- Is it possible, to open video files, which are not supported by MSU VQMT directly (.mp4, .mpg, .mov etc.)?
- Why results are unequal (too low) for one or more codecs?
- What about masking? I really need it!
- Your masking is not comfortable; I want to draw a mask on a video frame!
- Why MSU VQMT has four variants of PSNR calculation?
- What about OPSNR? Why MSU VQMT does not have it?
- Why PSNR is slightly different from previous version?
- Why MSU VQMT has two variants of SSIM calculation?
- Why neither of your SSIM calculations does match AviSynth SSIM plugin?
- Why SSIM (fast) visualization seems to be shifted?
- Why MSU Noise Estimation Metric plugin does not match to its previous version?
- Why MSU Noise Estimation Metric plugin does not match to VirtualDub MSU Noise Estimation filter?
- How can I obtain version for Linux?

How can I perform job processing with MSU VQMT?

You can make batch files to handle your jobs. The examples are given in the "Batch files examples" paragraph.

Your program reports that it failed to open input file.

Our tool supports .avi, .yuv, .avs, .bmp, .y4m files as input. Other file types (.wmv, .vob, .mpeg, etc.) can be opened through AviSynth. Our GUI tool can generate the necessary AviSynth script automatically. For more information read AviSynth Introduction paragraph in GUI help.

If you experience a problem with .avi, then it is possible that MSU VQMT is not compatible with this codec. Please send us information about this codec, or part of such avi file (1-2 megabytes) by e-mail.
What’s about values of the metrics? What values for each metrics mean, that quality is better?

<table>
<thead>
<tr>
<th>Metric</th>
<th>Full Reference (FR)/No Reference (NR)</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR, PSNR (256), APSNR, APSNR (256)</td>
<td>FR</td>
<td>100 for equal frames, higher values are better</td>
</tr>
<tr>
<td>SSIM (fast), SSIM (precise), MS-SSIM (fast), MS-SSIM (precise), 3SSIM, stSSIM</td>
<td>FR</td>
<td>Higher values are better, 1 for equal frames</td>
</tr>
<tr>
<td>VQM</td>
<td>FR</td>
<td>0 for equal frames, lower values are better</td>
</tr>
<tr>
<td>MSU Blocking metric</td>
<td>NR</td>
<td>Lower value corresponds to lower blocking.</td>
</tr>
<tr>
<td>MSU Blurring metric</td>
<td>NR</td>
<td>Lower value corresponds to higher blurring.</td>
</tr>
<tr>
<td>Delta</td>
<td>FR</td>
<td>0 means equal frames, positive and negative values mean deviation, lower absolute values are better</td>
</tr>
<tr>
<td>MSAD</td>
<td>FR</td>
<td>0 means equal frames, lower values are better</td>
</tr>
<tr>
<td>MSE</td>
<td>FR</td>
<td>Lower values are better, 0 for equal frames</td>
</tr>
<tr>
<td>MSU Brightness Flicking Metric</td>
<td>NR</td>
<td>The higher the value, the higher the brightness flicking in comparison with the previous frame</td>
</tr>
<tr>
<td>MSU Brightness Independent PSNR</td>
<td>FR</td>
<td>100 for equal frames, higher values are better</td>
</tr>
<tr>
<td>MSU Drop Frames Metric</td>
<td>NR</td>
<td>Metric has two values: 0 means that current frame exists, 1 that it is dropped</td>
</tr>
<tr>
<td>MSU Noise Estimation Metric</td>
<td>NR</td>
<td>The higher the value, the higher the noise level</td>
</tr>
<tr>
<td>MSU Scene Change Detector</td>
<td>NR</td>
<td>Metric has two values: 1 means that current frame is the first frame in a scene, for other frames metric value is 0.</td>
</tr>
</tbody>
</table>

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 (http://www.compression.ru/video/quality_measure/vqmt_sdk_en.html#start) 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. There is also known difference between AviSynth and VirtualDub.

Is it possible to open video files, which are not supported by MSU VQMT directly (.mp4, .mpg, .mov etc.)?

Our program supports AviSynth, it is possible to open such files using that software. Most of files you can open using scripts auto generation. If it is not working, try to write AviSynth script on your own.

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. 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.

User can specify mask color. 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:

- We assume the color with three zero components in RGB and with zero Y-component in YUV as “black”
- If, for example, the mask file is provided in RGB and calculations are performed in YUV, then mask color is converted into YUV using provided conversion settings and the masked areas are filled with the color achieved
- 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 comfortable; 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:

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

"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:

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

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:

<table>
<thead>
<tr>
<th>Metric</th>
<th>MaxErr calculation</th>
<th>Average PSNR calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>correct</td>
<td>correct</td>
</tr>
<tr>
<td>PSNR (256)</td>
<td>256 (fast, inexact)</td>
<td>correct</td>
</tr>
<tr>
<td>APSNR</td>
<td>correct</td>
<td>averaging</td>
</tr>
<tr>
<td>APSNR (256)</td>
<td>256 (fast, inexact)</td>
<td>averaging</td>
</tr>
</tbody>
</table>

"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 comfortable; 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 a 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 Gauss 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. But 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 originates from 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?
Sorry, our program is Windows-only for now.

List of figures

Picture 1. Delta example for two frames ...................................................... 24
Picture 2. Original and processed images (for Delta example) ...................... 25
Picture 3. Delta values for original and processed images (for Delta example) 26
Picture 4. MSE example for two frames ....................................................... 27
Picture 5. Original and processed images (for MSE example) ....................... 28
Picture 6. MSE values for original and processed images (for MSE example) ... 29
Picture 7. MSAD example for two frames ..................................................... 30
Picture 8. Original and processed images (for MSAD example) ..................... 31
Picture 9. MSAD values for original and processed images (for MSAD example) 32
Picture 10. PSNR example for two frames ..................................................... 35
Picture 11. Original and processed images (for PSNR example) .................... 36
Picture 12. PSNR values for original and processed images (for PSNR example) 37
Picture 13. PSNR Example for RD-curve ...................................................... 38
Picture 14. Diagram of the structural similarity (SSIM) measurement system .... 39
Picture 15. SSIM example for compressed video ......................................... 42
Picture 16. Original and processed video sequences (for SSIM example) ... ... 43
Picture 17. SSIM (fast) visualizations for original and processed video sequences 44
Picture 18. SSIM (precise) visualizations for original and processed video sequences 45
Picture 19. SSIM Example for RD-curve ...................................................... 46
Picture 20. VQM example for processed image ............................................ 65
Picture 21. Original and processed images (for VQM example) .................... 66
Picture 22. VQM values for original and processed images (for VQM example) 67
Picture 23. VQM Example for RD-curve ...................................................... 68
Picture 24. Used for blurring metrics calculation pixels ................................ 69
Picture 25. Examples of blurring metrics ..................................................... 70
Picture 26. Original and processed images (for MSU Blurring example) ........... 71
Picture 27. MSU Blurring values for original and processed images ............... 72
Picture 28. MSU Blurring example ............................................................. 73
Picture 29. Pixels, used for blocking metrics calculation ................................ 74
Picture 30. Geometric sense of D for MSU Blocking ...................................... 75
Picture 31. Shape of function W(x) for MSU Blocking ................................... 76
Picture 32. A. Decoded frame. B. Visualization of blocking metric ................. 76
Picture 33. Original and compressed images (for MSU Blocking example) ... 77
Picture 34. MSU Blocking values for original and processed images ............... 77
Picture 35. MSU Blocking example ............................................................. 78
Picture 36. MSU Brightness Flicking Metric, per-frame visualization ............... 88
Picture 37. MSU Brightness Independent PSNR visualization ......................... 90
Picture 38. MSU Drop Frame Metric visualization ......................................... 91
Picture 39. MSU Noise Estimation Metrics (Block-based, STG and MAD) visualization 93
Picture 40. MSU Scene Change Visualization .............................................. 94
Picture 41. MSU Scene Change Detector plot .............................................. 94
Picture 42. Example of per-frame results visualization using MATLAB .......... 106
Picture 43. Example of per-frame results visualization using MATLAB .......... 107
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.

REC.601 Table

<table>
<thead>
<tr>
<th>RGB to YUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16</td>
</tr>
<tr>
<td>U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128</td>
</tr>
<tr>
<td>V = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>YUV to RGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = 1.164 * (Y - 16) + 1.596 * (V - 128)</td>
</tr>
<tr>
<td>G = 1.164 * (Y - 16) - 0.391 * (U - 128) - 0.813 * (V - 128)</td>
</tr>
<tr>
<td>B = 1.164 * (Y - 16) + 2.018 * (U - 128)</td>
</tr>
</tbody>
</table>

PC.601 Table

<table>
<thead>
<tr>
<th>RGB to YUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = 0.299 * R + 0.587 * G + 0.114 * B</td>
</tr>
</tbody>
</table>

U = -(0.147) * R - 0.289 * G + 0.436 * B
V = 0.615 * R - 0.515 * G - 0.100 * B

**YUV to RGB**

R = Y + 1.14 * V
G = Y - 0.395 * U - 0.581 * V
B = Y + 2.032 * U

**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 describes metric and color component, used for comparison (PSNR, SSIM for Y component in YUV color space in the example below). Next two lines contain names of the compared files. For metrics, which use only one file for comparison, third line is blank. Next line (AVG) contains average value of the metric for whole sequence. After these four lines each line contains value of the metric for correspondent frames (5th line for 1st frame, 6th line for 2nd frame etc.).

Structure of CSV file using regular expressions

Structure of the CVS file can be described using the following regular expressions:

| Metric, color component | (<METRIC_NAME><D>)*<METRIC_NAME>
| Name of the first file | (<FILE_NAME1><D>)* <FILE_NAME1>
| Name of the second file | (<FILE_NAME2><D>)* <FILE_NAME2>
| Average values | (AVG:\s<AVG_NUMBER><D>)*AVG:\s<AVG_NUMBER>
| Values for 1st frame | (<METRIC_VALUE><D>)* <METRIC_VALUE>
| ... | ...
| Values for Nth frame | (<METRIC_VALUE><D>)* <METRIC_VALUE>

Where <D> is defined CSV delimiter ("", "", or ";").

Example of CSV file

| Metric, color component | PSNR_YYUV | PSNR_YYUV | SSIM_YYUV | SSIM_YYUV |
| Name of the first file | foreman.avi | foreman.avi | foreman.avi | foreman.avi |
| Name of the second file | foreman_100.avi | foreman_200.avi | foreman_100.avi | foreman_200.avi |
| Average value | AVG: 31.20652 AVG: 33.72194 AVG: 0.84871 AVG: 0.89998 |
| Value for 1st frames | 40,33786 40,33786 0.96335 0.96335 |
| Value for 2nd frames | 38,25759 38,25759 0.95292 0.95292 |
| Value for 3rd frames | 38,36161 38,36161 0.95278 0.95278 |
| ... | 37,64412 38,02011 0.94651 0.94876 |
| ... | 37,31698 37,84814 0.9436 0.94788 |
| ... | 36,78943 37,70628 0.93588 0.9434 |
| ... | 36,68823 37,10873 0.94042 0.94404 |
| ... | 36,70002 37,18142 0.94015 0.94385 |
| ... | 35,68851 36,27249 0.92219 0.92909 |
| ... | 35,58341 36,63955 0.93059 0.94055 |
| ... | 35,51961 36,6788 0.92853 0.93945 |
| ... | 34,8146 36,97665 0.91205 0.93483 |
Average CSV file

In this file each line corresponds to one video file which is compared with the original. So, if you have made 10 comparisons with the same “-af” parameter, 10 lines will be appended to this file.

The first value of each line is a name of the processed video sequence. After that values of metrics (average for the whole video) are placed.

Structure of CSV file using regular expressions

The structure of an average CSV file can be described using the following regular expressions:

<table>
<thead>
<tr>
<th>List of metrics &amp; color spaces</th>
<th>VIDEO&lt;D&gt;(&lt;METRIC_NAME&gt;&lt;D&gt;)*&lt;METRIC_NAME&gt;\n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average values for 1st file</td>
<td>&lt;FILE_NAME2&gt;&lt;D&gt;(&lt;AVG_VALUE&gt;&lt;D&gt;)*&lt;AVG_VALUE&gt;\n</td>
</tr>
<tr>
<td>Average values for Nth file</td>
<td>&lt;FILE_NAME2&gt;&lt;D&gt;(&lt;AVG_VALUE&gt;&lt;D&gt;)*&lt;AVG_VALUE&gt;\n</td>
</tr>
</tbody>
</table>

Where <D> is the defined CSV delimiter ("","",":").

Example of average CSV file

<table>
<thead>
<tr>
<th>List of metrics &amp; color spaces</th>
<th>VIDEO</th>
<th>PSNR_YYUV</th>
<th>SSIM_YYUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average values for 1st file</td>
<td>D:\Video\Samples\foreman_100.avi</td>
<td>31,20652</td>
<td>0,84871</td>
</tr>
<tr>
<td>Average values for 2nd file</td>
<td>D:\Video\Samples\foreman_200.avi</td>
<td>33,72194</td>
<td>0,89998</td>
</tr>
<tr>
<td>Average values for 3rd file</td>
<td>D:\Video\Samples\foreman_300.avi</td>
<td>35,45608</td>
<td>0,92568</td>
</tr>
<tr>
<td>...</td>
<td>D:\Video\Samples\foreman_400.avi</td>
<td>36,57669</td>
<td>0,93998</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_500.avi</td>
<td>37,4851</td>
<td>0,94957</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_600.avi</td>
<td>38,15945</td>
<td>0,95621</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_700.avi</td>
<td>38,75105</td>
<td>0,96082</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_800.avi</td>
<td>39,28659</td>
<td>0,96495</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_900.avi</td>
<td>39,88145</td>
<td>0,96869</td>
</tr>
<tr>
<td></td>
<td>D:\Video\Samples\foreman_1000.avi</td>
<td>40,16985</td>
<td>0,97061</td>
</tr>
</tbody>
</table>

Average CSV file, second version

This new version of average CSV files is implemented because the previous version has weak handling of no reference metrics, because it does not take into account that we may need to store metric values for original file. All the metric values that correspond only to the original file (no reference metrics) are not stored in the previous version.

In this version of the average CSV file structure each line corresponds to the set of measured videos. As for the previous version, if you make 10 comparisons with the same “-af2” parameter, 10 lines will be appended to the given file.

First two values of each line are file names of video sequences the metrics were calculated for. Next, values of the metrics (average for the whole video sequence) are placed. For no reference metrics the second file name is replaced with “-” sign. Also the values of the metrics that were not calculated for the current file set are replaced by “-” sign.

Structure of CSV file using regular expressions

Structure of an average CVS file can be described using the following regular expressions:

<table>
<thead>
<tr>
<th>List of metrics &amp; color spaces</th>
<th>video1&lt;D&gt;video2&lt;D&gt;({&lt;METRIC_NAME&gt;&lt;D&gt;})*&lt;METRIC_NAME&gt;\n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average values for 1st file</td>
<td>&lt;FILE_NAME1&gt;&lt;D&gt;&lt;FILE_NAME2&gt;&lt;D&gt;({&lt;AVG_VALUE&gt;&lt;D&gt;})*&lt;AVG_VALUE&gt;\n</td>
</tr>
<tr>
<td>Average values for Nth file</td>
<td>&lt;FILE_NAME1&gt;&lt;D&gt;&lt;FILE_NAME2&gt;&lt;D&gt;({&lt;AVG_VALUE&gt;&lt;D&gt;})*&lt;AVG_VALUE&gt;\n</td>
</tr>
</tbody>
</table>

Where <D> is the defined CSV delimiter (“,” or “;”).

Example of average CSV file

<table>
<thead>
<tr>
<th>List of metrics &amp; color spaces</th>
<th>video1</th>
<th>video2</th>
<th>PSNR_YYUV</th>
<th>MSU Blurring BETA_YYUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>bus_o.avi</td>
<td>bus_c.avi</td>
<td>19.94899</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>bus_o.avi</td>
<td>-</td>
<td>-</td>
<td>27.89646</td>
<td></td>
</tr>
<tr>
<td>bus_c.avi</td>
<td>-</td>
<td>-</td>
<td>27.36283</td>
<td></td>
</tr>
</tbody>
</table>
Parsing in MATLAB

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

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

Picture 63 show results of this script execution.

![Example of per-frame results visualization using MATLAB](http://www.compression.ru/video/quality_measure/vqmt_pro_en.html)
Parsing of the average results file can be performed with the same way:

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

Picture 64 show results of this script execution.

![Average data visualization](image)

**Picture 64.** Example of per-frame results visualization using MATLAB
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