ISO/IEC JTC 1/SC 29/WG 04
MPEG Video Coding
Convenorship: CN

Document type:	Output Document
Title:	Software manual of QMIV
Status:	Approved
Date of document:	2024-07-26
Source:	ISO/IEC JTC 1/SC 29/WG 04
Expected action:	None
Action due date:	None
No. of pages:	13 (without cover page)
Email of Convenor:	yul@zju.edu.cn
Committee URL:	https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-4

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 04 MPEG VIDEO CODING

ISO/IEC JTC 1/SC 29/WG 04 N 0535

July 2024, Sapporo, JP

Title	**Software manual of QMIV**
Source	**WG 04 MPEG Video Coding**
Status	**Approved**
Serial Number	**24266**
Authors	**Jakub Stankowski, Adrian Dziembowski**

1. Introduction

This document describes a successor of the IVPSNR software: QMIV. The QMIV framework includes additional metrics based on structural similarity: SSIM and IV-SSIM. QMIV v1.0 calculates: PSNR, WS-PSNR [Sun17], IV-PSNR [Dziembo22], SSIM [Wang04], and IV-SSIM [M68223].

All the metrics are calculated separately for each frame of the sequence. In the end, the QMIV framework returns the mean value of each metric, averaged over a desired number of frames.

1.1. PSNR

PSNR for each color component c is calculated as:

$$\mathrm{PSNR}(c) = 10 \cdot \log\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}(c)}\right),$$

where MAX is the maximum value of the color component (e.g., 1023 for 10-bit video) and:

$$\mathrm{MSE}(c) = \frac{1}{W \cdot H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \left(c_R(x,y,c) - c_T(x,y,c)\right)^2 ,$$

where W and H are width and height of the image, $c_T(x,y,c)$ and $c_R(x_R,y_R,c)$ are values of color component c in the position (x,y) in the test image and the reference image, respectively.

The QMIV framework returns the PSNR value for each color component c (by default: Y, Cb, and Cr), as well as the weighted average of those values, calculated with weights determined by the **CmpWeightsAverage** parameter.

The QMIV tries to avoid inconsistency caused by divide-by-zero issue in case of perfect frames. If there is completely no distortion the $\mathrm{MSE}(c) = 0$ leads to $\mathrm{PSNR}(c) = \infty$. Direct averaging of such PSNR values leads to average PSNR also equal to infinity. This makes comparison of very high-quality sequences almost impossible since even one perfect frame out of thousands will set the final PSNR to +inf. To avoid this issue, we introduced “fake PSNR” mode for perfect frames. In this case the PSNR metric for undistorted component is set to:

$$\mathrm{PSNR}_{FAKE}(c) = 10 \cdot \log\left(\frac{\mathrm{MAX}^2}{1}\right)$$

This behavior mimics a situation where one pixel differs by one. In order to avoid a silent altering of metric value, this mode is signaled in output log and result file. For every undistorted component of each frame, the software emits `”Exact{CmpName}”` message.

1.2.WS-PSNR [Sun17]

WS-PSNR is a PSNR-based metric adapted for omnidirectional video. In [Sun17] handling of different projections is described. In the QMIV framework, only the ERP projection handling is implemented.

The WS-PSNR value for each color component c is calculated as:

$$\mathrm{WSPSNR}(c) = 10 \cdot \log\left(\frac{\mathrm{MAX}^2}{\mathrm{WSMSE}(c)}\right),$$

where MAX is the maximum value of the color component (e.g., 1023 for 10-bit video) and:

$$\mathrm{WSMSE}(c) = \frac{\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\left(c_R(x,y,c) - c_T(x,y,c)\right)^2 \cdot w_{x,y}}{\sum_{y=0}^{H-1}\sum_{x=0}^{W-1} w_{x,y}},$$

where W and H are width and height of the image, $c_T(x,y,c)$ and $c_R(x_R,y_R,c)$ are values of color component c in the position (x,y) in the test image and the reference image, respectively, and weight $w_{x,y}$ is calculated as:

$$w_{x,y} = \cos\frac{\left(y + 0.5 - \frac{H}{2}\right) \cdot \pi}{H}.$$

The QMIV framework returns the WS-PSNR value for each color component c (by default: Y, Cb, and Cr), as well as the weighted average of those values, calculated with weights determined by the **CmpWeightsAverage** parameter.

The WS-PSNR$(c) = \infty$ problem is handled in the same way as for PSNR metric.

1.3. IV-PSNR [Dziembo22]

IV-PSNR is a PSNR-based objective quality metric adapted for Immersive Video applications. Compared to PSNR, two major modifications were added: Corresponding Pixel Shift and Global Color Difference. Corresponding Pixel Shift eliminates the influence of a slight shift of objects’ edges caused by reprojection errors. Global Color Difference reduces the influence of different color characteristics of different input views.

Detailed description of the IV-PSNR metric can be found in [Dziembo22]. Below, the general and simplified idea of the IV-PSNR is presented.

IV-PSNR for color component c is calculated as:

$$\mathrm{IVPSNR}(c) = 10 \cdot \log\left(\frac{\mathrm{MAX}^2}{\mathrm{IVMSE}(c)}\right),$$

where MAX is the maximum value of the color component (e.g., 1023 for 10-bit video) and:

$$\mathrm{IVMSE}(c) = \frac{1}{W \cdot H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \min_{\substack{x\prime \in [x-\mathrm{CPS}, x+\mathrm{CPS}] \\ y\prime \in [y-\mathrm{CPS}, y+\mathrm{CPS}]}} \left(c_T(x,y,c) - c_R(x',y',c) + \mathrm{GCD(c)}\right)^2 \ ,$$

where W and H are width and height of the image, $c_T(x,y,c)$ and $c_R(x',y',c)$ are values of color component c in the position (x,y) in the test image and (x',y') in the reference image, respectively. CPS is the maximum Corresponding Pixel Shift between reference and test image, and GCD is the Global Color Difference for component c:

$$\mathrm{GCD}(c) = \max\left(\frac{1}{W \cdot H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \left(c_R(x,y,c) - c_T(x,y,c)\right) \ , \mathrm{MUD}(c)\right),$$

where $\mathrm{MUD}(c)$ is the Maximum Unnoticeable Difference for color component c. Position (x',y') within test image is determined by searching for the most similar pixel to the pixel in position (x,y) within the reference image.

In order to provide better quality assessment for omnidirectional video provided in the ERP format, weighting technique from WS-PSNR [Sun17] was applied.

The IV-PSNR quality metric is based on PSNR, therefore, the higher the number, the better is the quality.

The QMIV framework returns a single IV-PSNR value, calculated by weighting $\mathrm{IVPSNR}(c)$ for three color components c using weights determined by the CmpWeightsAverage parameter.

1.4.SSIM [Wang04]

The SSIM metric is based on comparing the structural similarity between two images. SSIM for color component c is calculated as:

$$\mathrm{SSIM}(c) = \frac{1}{W \cdot H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} L(x,y,c) \cdot C(x,y,c) \cdot S(x,y,c) \ ,$$

where W and H are width and height of the image, and:

$$L(x,y,c) = \frac{2 \cdot \mu_R(x,y,c) \cdot [\mu_T(x,y,c)] + C_1}{\mu_R(x,y,c)^2 + [\mu_T(x,y,c)]^2 + C_1},$$

$$C(x,y,c) = \frac{2 \cdot \sigma_R(x,y,c) \cdot \sigma_T(x,y,c) + C_2}{\sigma_R(x,y,c)^2 + \sigma_T(x,y,c)^2 + C_2},$$

$$S(x,y,c) = \frac{\sigma_{RT}(x,y,c) + C_3}{\sigma_R(x,y,c) \cdot \sigma_c^J(x,y,c) + C_3},$$

where C_1, C_2, and C_3 are constants providing numerical stability, and local statistics of both compared images are calculated as:

$$\mu_R(x,y,c) = \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} [\boldsymbol{\omega}(i-x, j-y) \cdot c_R(i,j,c)] \ ,$$

$$\mu_T(x,y,c) = \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} [\boldsymbol{\omega}(i-x, j-y) \cdot c_T(i,j,c)] \ ,$$

$$\sigma_R(x,y,c) = \sqrt{\sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \left[\boldsymbol{\omega}(i-x,j-y) \cdot \left(c_R(i,j,c)\right)^2\right] - \left(\mu_R(x,y,c)\right)^2} \,,$$

$$\sigma_T(x,y,c) = \sqrt{\sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \left[\boldsymbol{\omega}(i-x,j-y) \cdot \left(c_T(i,j,c)\right)^2\right] - \left(\mu_T(x,y,c)\right)^2} \,,$$

$$\sigma_{RT}(x,y,c) = \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} [\boldsymbol{\omega}(i-x,j-y) \cdot c_R(i,j,c) \cdot c_T(i,j,c)] - \mu_R(x,y,c) \cdot \mu_T(x,y,c) \,,$$

where $\boldsymbol{\omega}$ is the weighting mask of size $2k+1 \times 2k+1$, while $c_T(x,y,c)$ and $c_R(x_R,y_R,c)$ are values of color component c in the position (x,y) in the test image and the reference image.

The QMIV framework returns the SSIM value for each color component c (by default: Y, Cb, and Cr), as well as the weighted average of those values, calculated with weights determined by the **CmpWeightsAverage** parameter.

1.5. IV-SSIM [M68223]

IV-SSIM is a SSIM-based objective quality metric adapted for Immersive Video applications. Compared to SSIM, two major modifications (the same as for IV-PSNR) were added: Corresponding Pixel Shift and Global Color Difference.

Detailed description of the IV-SSIM metric can be found in [M68223]. Below, the general and simplified idea of the IV-SSIM is presented.

IV-SSIM for color component c is calculated as:

$$\mathrm{IVSSIM}(c) = \frac{1}{W \cdot H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} L(x,y,c) \cdot C(x,y,c) \cdot S(x,y,c) \,,$$

where W and H are width and height of the image, and:

$$L(x,y,c) = \frac{2 \cdot \mu_R(x,y,c) \cdot [\mu_T(x,y,c) + \mathrm{GCD}(c)] + C_1}{\mu_R(x,y,c)^2 + [\mu_T(x,y,c) + \mathrm{GCD}(c)]^2 + C_1} \,,$$

where $\mathrm{GCD}(c)$ is calculated in the same way as for IV-PSNR. $C(x,y,c)$ and $S(x,y,c)$, as well as local statistics for reference image (σ_R and μ_R) are calculated using the same equations, as for SSIM. Other three local statistics are calculated as:

$$\mu_T(x,y,c) = \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} [\boldsymbol{\omega}(i-x,j-y) \cdot c_T(i',j',c)] \,,$$

$$\sigma_T(x,y,c) = \sqrt{\sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \left[\boldsymbol{\omega}(i-x,j-y) \cdot \left(c_T(i',j',c)\right)^2\right] - \left(\mu_T(x,y,c)\right)^2} \,,$$

$$\sigma_{RT}(x,y,c) = \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} [\boldsymbol{\omega}(i-x,j-y) \cdot c_R(i,j,c) \cdot c_T(i',j',c)] - \mu_R(x,y,c) \cdot \mu_T(x,y,c) \,,$$

where the position (i',j') is calculated by analyzing the $B \times B$ neighborhood of the pixel (i,j) within image c_T in search of the pixel, which is most similar to the pixel (i,j) in image c_R.

In order to provide better quality assessment for omnidirectional video provided in the ERP format, weighting technique from WS-PSNR [Sun17] was applied.

The IV-SSIM quality metric is based on SSIM, therefore, the higher the number, the better is the quality.

The QMIV framework returns a single IV-SSIM value, calculated by weighting $\text{IVSSIM}(c)$ for three color components c using weights determined by the **CmpWeightsAverage** parameter.

2. Software manual

QMIV v1.0 accepts commandline parameters listed in section 2.1:

2.1. Commandline parameters

General parameters

Cmd	ParamName	Description
-i0	InputFile0	File path - input sequence 0
-i1	InputFile1	File path - input sequence 1
-ff	FileFormat	Format of input sequence (optional, default: RAW) [RAW, PNG]
-ps	PictureSize	Size of input sequences (WxH e.g., 1920x1080) [1]
-pw	PictureWidth	Width of input sequence [1]
-ph	PictureHeight	Height of input sequence [1]
-pf	PictureFormat	Picture format, as defined by FFMPEG pix_fmt (e.g., yuv420p10le) [2]
-bd	BitDepth	Bit depth (optional, default: 8, up to 14) [2]
-cf	ChromaFormat	Chroma format (optional, default: 420) [420, 444] [2]
-s0	StartFrame0	Start frame (optional, default: 0)
-s1	StartFrame1	Start frame (optional, default: 0)
-nf	NumberOfFrames	Number of frames to be processed (optional, default: -1 = all)
-r	ResultFile	Output file path for printing results (optional)
-ml	MetricList	List of quality metrics to be calculated, must be coma separated, quotes are required. "All" enables all available metrics. (optional, default="PSNR, WSPSNR, IVPSNR, IVSSIM") [IVPSNR, IVSSIM, PSNR, SSIM, WSPSNR]

[1] PictureSize parameter can be used interchangeably with PictureWidth, PictureHeight pair; if PictureSize parameter is present the PictureWidth and PictureHeight arguments are ignored.
[2] PictureFormat parameter can be used interchangeably with BitDepth, ChromaFormat pair; if PictureFormat parameter is present the BitDepth and, ChromaFormat arguments are ignored.

Masked mode parameters

Cmd	ParamName	Description
-im	InputFileM	File path – mask (optional, same resolution as InputFile0 and InputFile1)
-bdm	BitDepthM	Bit depth for mask (optional, default: BitDepth, up to 16)
-cfm	ChromaFormatM	Chroma format for mask (optional, default: ChromaFormat) [400, 420, 444]

Equirectangular parameters

Cmd	ParamName	Description
-erp	Equirectangular	Equirectangular sequence (flag, default disabled)
-lor	LonRangeDeg	Longitudinal range of ERP sequence [°] (optional, default: 360)
-lar	LatRangeDeg	Lateral range of ERP sequence [°] (optional, default: 180)

Colorspace mode parameters

Cmd	ParamName	Description
-csi	ColorSpaceInput	Color space of input files (optional, default: YCbCr)
-csm	ColorSpaceMetric	Color space used to calculate metrics (optional, default: ColorSpaceInput) If ColorSpaceInput != ColorSpaceMetric the software performs on-demand conversion (RGB-->YCbCr or YCbCr-->RGB). Conversion requires specific YCbCr color space parameters. [RGB, BGR, YCbCr, YCbCr_BT601, YCbCr_SMPTE170M, YCbCr_BT709, YCbCr_SMPTE240M, YCbCr_BT2020]

IV-metric specific parameters

Cmd	ParamName	Description
-sr	SearchRange	IV-metric search range around center point (optional, default: 2 → 5×5)
-cws	CmpWeightsSearch	IV-metric component weights used during search ("Lm:Cb:Cr:0" or "R:G:B:0" - per component integer weights, default="4:1:1:0", quotes are mandatory, requires USE_RUNTIME_CMPWEIGHTS=1)
-cwa	CmpWeightsAverage	IV-metric component weights used during averaging ("Lm:Cb:Cr:0" or "R:G:B:0" - per component integer weights, default="4:1:1:0", quotes are mandatory)
-unc	UnnoticeableCoef	IV-metric unnoticable color difference threshold coeff ("Lm:Cb:Cr:0" or "R:G:B:0" - per component coeff, default="0.01:0.01:0.01:0", quotes are mandatory)

Validation parameters

Cmd	ParamName	Description
-ipa	InvalidPelActn	Select action taken if invalid pixel value (larger than [(1<<BitDepth)-1]) is detected (optional, default STOP) [SKIP – disable pixel value checking, WARN – print warning and ignore, STOP – stop execution, CNCL – try to conceal by clipping to bit depth range]
-nma	NameMismatchActn	Select action taken if parameters derived from filename are different than provided as input parameters. Checks resolution, bit depth and chroma format. (optional, default WARN) [SKIP – disable checking, WARN – print warning and ignore, STOP – stop execution]

Application parameters

Cmd	ParamName	Description
-nth	NumberOfThreads	Number of worker threads (optional, default: -2, suggested: ~8 for IV-PSNR, all physical cores for SSIM) [0 = thread pool disabled, -1 = all available threads, -2 = reasonable auto]
-ilp	InterleavedPic	Use additional image buffer with interleaved layout (improves performance at a cost of increased memory usage, optional, default: 1)
-v	VerboseLevel	Verbose level (optional, default: 1), cf. section 2.4

External config file

Cmd	ParamName	Description
-c		Valid path to external config file - in INI format (optional). Multiple config files can be provided by using multiple "-c" arguments. Config files are processed in arguments order. Content of config files are merged while repeating values are overwritten. cf. section 2.6

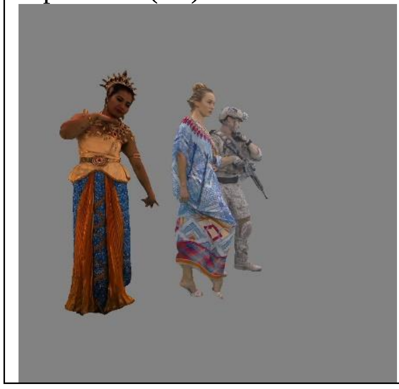
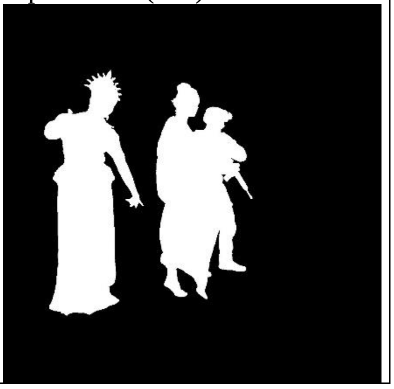
Dynamic dispatcher parameters

Cmd	Description
--DispatchForceMFL	Force dispatcher to selected microarchitecture (optional, default=UNDEFINED) [x86-64, x86-64-v2, x86-64-v3, x86-64-v4]. Forcing a selection of microarchitecture level not supported by CPU will lead to "illegal instruction" exception.
--DispatchVerbose	Verbose level for runtime dispatch module (optional, default=0).

- The commandline parameters are processed in arguments order.
- Multiple config files can be provided.
- When no parameters are used, syntax help is printed.

2.2. Masked mode

Optional mode of the QMIV allows to calculate the metrics only for specified areas. In order to use masked mode, InputFileM (-im) parameter has to be set, indicating a path of mask YUV file.

InputFile0 (-i0): 	InputFile1 (-i1):	InputFileM (-im): 

In an example above, the quality is calculated only for occupied pixels (as indicated by mask), so different color of the unoccupied background does not impact outputted quality.

Requirements and notes

- Resolution of mask file has to be identical as input file.
- Allowed mask values are 0 (interpreted as inactive pixel) and (1<<BitDepthM)-1) (interpreted as active pixel). Behavior for other values is undefined at this moment.
- The data processing functions for masked mode are not implemented with the use of SIMD instructions.
- Masked SSIM and IV-SSIM are not supported in v1.0.

2.3. Colorspace modes

Optional mode of the QMIV allows to define or convert the colorspace of input sequences and/or colorspace used to calculate metrics.

The default behavior is to assume generic YCbCr colorspace for both input and metric and perform no conversion. The metric values are decorated with YCbCr or Y:Cb:Cr suffix.

The second option is to use RGB input files and calculate metric in RGB. In this case the QMIV performs no conversion and output metrics labeled as RGB, i.e., PSNR R:G:B PSNR-RGB. To enable this mode use "-csi RGB" input argument.

When calculating metrics in RGB mode, component weights should be set equal for all three components:

- CmpWeightsAverage = "1:1:1:0",
- CmpWeightsSearch = "1:1:1:0".

This can be archived using config file or commandline (-cwa "1:1:1:0" -cws "1:1:1:0").

The other path assumes on-demand colorspace conversion. The software is able to perform RGB-to-YCbCr or YCbCr-to-RGB conversion. In this case, a specific variant of YCbCr colorspace (BT601, SMPTE170M, BT709, SMPTE240M, BT2020) have to be defined for input or metric. The valid arguments for YCbCr colorspace variants are [YCbCr_BT601, YCbCr_SMPTE170M, YCbCr_BT709, YCbCr_SMPTE240M, YCbCr_BT2020].

For example, if one wants to use input sequence in BT709 colorspace and calculate metrics in RGB, the following parameters should be used: -csi YCbCr_BT709 -csm RGB -cwa “1:1:1:0” -cws “1:1:1:0”.

2.4.PNG mode

Optional mode of the QMIV allows to calculate the metrics not only for video files, but for lists of indexed images in the PNG format.

The name of input file should contain format string as defined in C++20 std::format [ISO/IEC 14882:2020] in the form of “{:d}” with optional modifiers. The file name is determined by formatting input string using image index.

The software expects lists of consequently numbered files. The first file index can be equal to 0 or 1. The last file is detected by checking the existence of consecutive files. The first missing index is treated as the end of the file list.

Examples:

- The list of files {img001.png, img002.png, img003.png} should be specified as “img{:03d}.png”.
- The list of files {img000.png, img001.png, img002.png, img004.png}, specified as “img{:03d}.png”, will be processed for 0,1, and 2 indexes only. The “img002.png” will be detected as the last image in list.

2.5. Verbose levels

Value	Printed data
0	Final metrics values only
1	0 + configuration + detected number of frames
2	1 + argc/argv + frame level metric values
3	2 + computing time (could slightly slow down computations)
4	3 + QMIV specific debug data (GlobalColorShift, R2T+T2R, NumNonMasked)
9	stdout flood

2.6. Compile-time parameters

Parameter name	Default value	Description
USE_SIMD	1	use SIMD (to be precise... use SSE 4.1, AVX2, or AVX512)
USE_RUNTIME_CMPWEIGHTS	1	use component weights provided at runtime

2.7. Config file example

```
InputFile0       = "SA_ref.yuv"
InputFile1       = "SA_test.yuv"
PictureWidth     = 4096
PictureHeight    = 2048
BitDepth         = 10
```

```
ChromaFormat    = 420
VerboseLevel    = 3
MetricList      = "PSNR, IVPSNR, IVSSIM"
OutputFile      = "QMIV.txt"
```

2.8. Compilation requirements

The QMIV v1.0 software uses following external components:

- libfmt – Formatting library for C++ – distributed under BSD license and automatically fetched by QMIV CMake-based buid system,
- libspng – Simple PNG – C library for reading and writing Portable Network Graphics – distributed under MIT License
- miniz – Single C source file zlib-replacement library – distributed under BSD 2-Clause "Simplified" License

In order to build the software, the ISO C++17 conformant compiler is required.

3. Building

Building the QMIV software requires using CMake (https://cmake.org/) and C++17 conformant compiler (e.g., GCC >= 8.0, clang >= 5.0, MSVC >= 19.15).

The QMIV application and its build system is designed to create the fastest possible binary. On x86-64 microarchitectures the build system can create four version of compiled application, each optimized for one predefined x86-64 Microarchitecture Feature Levels [x86-64, x86-64-v2, x86-64-v3, x86-64-v4] (defined in https://gitlab.com/x86-psABIs/x86-64-ABI). The final binary consists of these four optimized variants and a runtime dynamic dispatcher. The dispatcher uses the CPUID instruction to detect available instruction set extensions and selects the fastest possible code path.

The QMIV CMake project defines the following parameters:

Variable	**Type**	**Description**
PMBB_GENERATE_MULTI _MICROARCH_LEVEL_BINARIES	BOOL	Enables generation of multiple code paths, optimized for each variant of x86-64 Microarchitecture Feature Levels.
PMBB_GENERATE_SINGLE_APP _WITH_WITH_RUNTIME_DISPATCH	BOOL	Enables building single application with runtime dynamic dispatch. Requires PMBB_GENERATE _MULTI_MICROARCH_LEVEL_BINARIES=True.
PMBB_GENERATE_DEDICATED _APPS_FOR_EVERY_MFL	BOOL	Enables building multiple applications, each optimized for selected x86-64 Microarchitecture Feature Level. Requires PMBB_GENERATE_MULTI_MICROARCH _LEVEL_BINARIES=True.
PMBB_BUILD_WITH_MARCH _NATIVE	BOOL	Enables option to force compiler to tune generated code for the micro-architecture and ISA extensions of the host CPU. Conflicts with PMBB_GENERATE_MULTI_MICROARCH_LEVEL _BINARIES. Generated binary is not portable across different microarchitecures.
PMBB_GENERATE_TESTING	BOOL	Enables generation of unit tests for critical data processing routines.

For user convenience, we prepared a set of scripts for easy "one click" configure and build:

- configure_and_build.bat - for Windows users

- configure_and_build.sh - for Unix/Linux users

For developer convenience, we prepared a set of scripts for easy "one click" configure in development mode - without multi-architecture build and dynamic dispatch:

- configure_for_developement.bat - for Windows users
- configure_for_developement.sh - for Unix/Linux users

4. Examples

1. Metrics (PSNR, WS-PSNR, IV-PSNR, and IV-SSIM) of SA_ref.yuv and SA_test.yuv. Sequence resolution is 4096×2048, YUV420, 10 bits per sample. Sequence format is ERP. Mean metrics calculated for the first 20 frames will be written into QMIV.txt:

```
QMIV -i0 SA_ref.yuv -i1 SA_test.yuv -pw 4096 -ph 2048 -bd 10 -erp -nf 20 -r QMIV.txt
```

or:

```
QMIV -i0 SA_ref.yuv -i1 SA_test.yuv -ps 4096x2048 -pf yuv420p10le -erp -l 20 -r QMIV.txt
```

2. Metrics (all available) of SD_ref.yuv and SD_test.yuv. Sequence resolution is 2048×1088, YUV420, 8 bits per sample. Sequence format is perspective. Mean metrics calculated for all frames will be written into results.txt:

```
QMIV -i0 SD_ref.yuv -i1 SD_test.yuv -r results.txt -pw 2048 -ph 1088 -ml “All”
```

3. Metrics (PSNR, WS-PSNR, and IV-PSNR) of SC_ref.yuv and SC_test.yuv. Sequence resolution is 4096×2048, YUV420, 10 bits per sample. Sequence format is ERP, with lateral range equal to 90°. Mean metrics calculated for 5 frames (frames 0-4 of reference video and 10-14 of test video) will be written into o.txt:

```
QMIV -i0 SC_ref.yuv -i1 SC_test.yuv -ps 4096x2048 -ml “PSNR, WSPSNR, IVPSNR” -r o.txt
     -erp -lar 90 -nf 5 -s1 10
```

4. Using config file:

```
QMIV -c "config.cfg"
```

5. Using external config file with some parameters added/overridden:

```
QMIV -c "config.cfg" -v 1 -nth 4
```

5. Software

MPEG Git Repository:	https://git.mpeg.expert/MPEG/Explorations/6DoF/qmiv
Public read-only access:	https://gitlab.com/mpeg-i-visual/qmiv
Software coordinator:	Jakub Stankowski, jakub.stankowski@put.poznan.pl

6. Usage and citation

Please cite reference [Dziembo22] when using IV-PSNR.

7. References

[Dziembo22] A. Dziembowski, D. Mieloch, J. Stankowski, A. Grzelka, “IV-PSNR – the objective quality metric for immersive video applications,” IEEE Transactions on Circuits and Systems for Video Technology 32 (11), 2022, DOI: 10.1109/TCSVT.2022.3179575.

[M48093] A. Dziembowski, M. Domański, “[MPEG-I Visual] Objective quality metric for immersive video,” ISO/IEC JTC1/SC29/WG11 MPEG/M48093, July 2019, Göteborg, Sweden.

[M54279] J. Stankowski, A. Dziembowski, “[MPEG-I Visual] Fast implementation of IV-PSNR software,” ISO/IEC JTC1/SC29/WG11 MPEG/M54279, July 2020, Online.

[M54896] J. Stankowski, A. Dziembowski, “Even faster implementation of IV-PSNR software,” ISO/IEC JTC1/SC29/WG04 MPEG VC/M54896, October 2020, Online.

[M55752] A. Dziembowski, J. Stankowski, “Slightly faster IVPSNR,” ISO/IEC JTC1/SC29/WG04 MPEG VC/M55752, January 2021, Online.

[M59974] J. Stankowski, A. Dziembowski, “Improved IV-PSNR software,” ISO/IEC JTC1/SC29/WG04 MPEG VC/M59974, July 2022, Online.

[M64727] J. Stankowski, A. Dziembowski, “Optimized IV-PSNR software with invalid pixel detection,” ISO/IEC JTC1/SC29/WG04 MPEG VC/M64727, Oct. 2023, Hannover, DE.

[M68222] J. Stankowski, A. Dziembowski, “The final version of the IV-PSNR software,” ISO/IEC JTC1/SC29/WG04 MPEG VC/M68222, Jul. 2024, Sapporo, JP.

[M68223] J. Stankowski, W. Nowak, A. Dziembowski, “IV-SSIM: adapting structural similarity to immersive video,” ISO/IEC JTC1/SC29/AG05 MPEG VQA/M68223, Jul. 2024, Sapporo, JP.

[Sun17] Y. Sun, A. Lu, L. Yu, “Weighted-to-Spherically-Uniform Quality Evaluation for Omnidirectional Video,” IEEE Signal Processing Letters 24 (9), pp. 1408-1412, 2017.

[Wang04] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, “Image quality assessment: From error measurement to structural similarity,” IEEE Transactions on Image Processing 13 (4), pp. 600-612, 2004.

8. Changelog

QMIV v1.0 [M68224]:

- introduction of Structural Similarity related metrics – SSIM and IV-SSIM,
- improved arguments parsing and validation,
- PictureSize argument introduced as (optional) alternative to PictureWidth, PictureHeight pair,
- PictureFormat argument introduced as (optional) alternative to BitDepth, ChromaFormat pair,
- improved consistency of selected "cmd" arguments:
    - PictureWidth ‘-w’ --> ’-pw’,
    - PictureHeight ‘-h’ --> ’-ph’,
    - NumberOfFrames ‘-l’ --> ‘-nf’,
    - ResultFile ‘-o’ --> ‘-r’,
    - NumberOfThreads ‘-t’ --> ‘-nth’,
- reduced overhead of computing time measurement (switched from ‘std::chrono:: high_resolution_clock’ to ‘RDTSCP’),
- increased precision of printed metric values,
- overhaul of IVPSNR v6.0 RGB mode into generic colorspace mode, including:
    - consistent metric suffixes,
    - RGB → YCbCr and YCbCr →RGB conversion,
    - RGB passthrough mode,
    - CalcMetricInRGB and ColorSpace arguments replaced by more general set of arguments (ColorSpaceInput, ColorSpaceMetric),
- input defined as PNG file or list of PNG files.

IVPSNR v6.0 [M64727]:

- yet another general overhaul of entire software structure,
- ComponentWeights parameter split into two new parameters CmpWeightsSearch and CmpWeightsAverage,
- added option to calculate metric in RGB color space:
    - the YCbCr input sequence is converted to RGB and metric is calculated on RGB pictures,
    - CalcMetricInRGB option added to enable this mode,
    - ColorSpace option added to select color space for YCbCr --> RGB conversion [BT601, SMPTE170M, BT709, SMPTE240M, BT2020],
- added fast SIMD implementation of Kahan-Babuška-Neumaier accumulation,
- added "reasonable auto" mode for number of threads selection and set as default behavior,
- added possibility to overwrite dynamic dispatcher decision and manually select code path,
- added weighted PSNR-YCbCr, WSPSNR-YCbCr, PSNR-RGB, and WSPSNR-RGB output.

IVPSNR v5.0 [M64727]:

- general overhaul of entire software structure,
- new cMake-based build system with simultaneous build of four variants of x86-64 Microarchitecture Feature Level and runtime dynamic dispatch,
- added unit tests for basic data processing routines,
- added detection invalid pel values (higher than (1<<BitDepth) - 1) and possibility to choose taken action (see InvalidPelAction parameter),
- added warning for settings influencing performance or breaking conformance with IV-PSNR metric defined in [M54279],
- added detection of mismatch between file name and provided parameters (resolution, bit depth and chroma format),
- added usage of hugepages on Linux-based systems (using madvise),
- added support for chroma format 4:2:2,
- more data processing functions implemented using AVX2,
- wider SIMD (AVX512) implementation for some data processing functions.

IVPSNR v4.0 [M59974]:

- SIMD (SSE 4.1) implementation of IV-PSNR calculation (for interleaved picture buffers),
- wider SIMD (AVX2) implementation for most data processing functions,
- runtime adjustable component weights for IV-PSNR metric,
- adjustable search range for IV-PSNR metric,
- adjustable unnoticeable color difference threshold coeff for IV-PSNR metric,
- reading parameters from config file,
- protection against StartFrame >= DetectedFrames,
- writing error messages to stdout and stderr,
- non-performance critical parameters moved from compile-time to run-time selection,
- added mask file option.

IVPSNR v3.0 [M55752]:

- enabled INTERPROCEDURAL_OPTIMIZATION and assumed x86-64 Microarchitecture Feature Level >= x86-64-v2,
- new implementation picture I/O,
- reduced filesystem burden (avoid repetitive open-seek-read-close cycles),
- use of interleaved picture layout for IVPSNR calculation,
- SIMD (SSE 4.1) implementation for most data processing functions,
- dedicated thread pool instead of OpenMP directives (due to high OpenMP overhead).

IVPSNR v2.1.1 (no reference):

- bug fixes.

IVPSNR v2.1 [M54896]:

- support for parallel processing (using OpenMP),
- addition of PSNR and WS-PSNR [Sun17] values outputting,
- fixed WS-weight calculation for ERP sequences with non-180 lateral range,
- changed commandline arguments formatting,
- addition of detection of corrupted YUV files,
- change in compile-time parameters:
  - VERBOSE_LEVEL is now a commandline parameter,
  - WSPSNR_PEAK_VALUE_8BIT flag added (default: enabled), when enabled, the signal peak value is set to 255 << (BitDepth – 8). Otherwise, it is equal to 2^BitDepth – 1.

IVPSNR v2.0 [M54279]:

- addition of (rOff) and (tOff) commandline parameters,
- removal of redundant GCD calculations,
- usage of uint16_t data type and 4:4:4 chroma format for internal picture storage,
- new implementation of pixel-level processing steps,
- reduction of filesystem burden by coalescing read,
- detection of read errors – causes application to exit returning EXIT_FAILURE,
- implementation of Kahanand-Babuska-Neumaier accumulation,
- improved conversion of 8bps input sequences,
- improved interpolation for input sequences with 4:2:0 chroma format,
- addition of 3 compile-time parameters:
  - VERBOSE_LEVEL – controls number of per-frame printing; default = 0,
  - USE_KBNS – enables the Kahan-Babuska-Neumaier accumulation; default: enabled,
  - USE_FIXED_WEIGHTS – enables faster 5×5 block search with fixed component weight (equal to 4:1:1); default = enabled,
- fixed possibility of reading from unallocated memory region during 5×5 block search,
- fixed GCD values rounding and clipping.

IVPSNR v1.0 [M48093].