MVC Software Manual

Version: JMVC 8.5 (CVS tag: JMVC_8_5)

Last update: March 26, 2011

Summary:

This document contains a detailed description of the usage and configuration of the JMVC (Joint Multiview Video Coding) software for the Multiview Video Coding (MVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG).

It provides information how to build the software on Windows32 and Linux platforms. It contains a description of the usage and configuration for the binaries built from the software package, including examples for multiview coding scenarios.

Table of Contents

1 General Information

The JMVC (Joint Multiview Video Coding) software is the reference software for the Multiview Video Coding (MVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG). Since the MVC project is still under development, the JMVC Software as is also under development and changes frequently.

The JMVC software is written in C++ and is provided as source code. Section 1.1 describes how the JMVC software can be obtained via a CVS server. Information about the structure of the CVS repository is presented in section 1.2. Section 1.3 describes how the JMVC software can be build on Win32 and Linux platforms, and section gives basic information about the binaries that are contained in the JMVC software package.

1.1 Accessing the latest JMVC Software

In order to keep track of the changes in software development and to always provide an up-to-date version of the JMVC software, a CVS server for the JMVC software has been set up at the Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen. The CVS server can be accessed using WinCVS or any other CVS client. The server is configured to allow read access only using the parameters specified in Table 1. Write access to the JMVC software server is restricted to the JMVC software coordinators group.

Table 1: CVS access parameters

authentication:	pserver
host address:	garcon.ient.rwth-aachen.de
path:	/cvs/jvt
user name:	jvtuser
password:	jvt.Amd.2
module name:	jmvc

Example 1 shows how the JMVC software can be accessed by using a command line CVS client.

Example 1: Accessing the JMVC software with a command line CVS client

```
cvs -d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
cvs -d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt checkout jmvc
```

In Example 2, it is shown how a specific JMVC software version – specified by a tag (JMVC_2_1 in Example 2) – can be obtained using a command line CVS client. Note that *co* represents an abbreviation for the command *checkout*, which was used in Example 1.

Example 2: Accessing the JMVC software version with the tag JMVC_2_1 with a command line CVS client

```
cvs -d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
cvs -d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt co -r JMVC_2_1 jmvc
```

1.2 Structure of the CVS Repository

After accessing the JMVC software as described in section 1.1, a folder *JMVC* is created. The directory structure of this folder is summarized in Table 2. Note that the folders *bin* and *lib* are created during building the software as described in section 1.3. The folder *JMVC* contains all files that are required for building and running the software.

The folder *JMVC/H264Extension/src* is structured into sub-folders for libraries, test projects, and tools. It contains all source and include files for the JMVC software, with exception of the include files that need to be shared by different libraries and/or test projects. These include files are located in the folder *JMVC/H264Extension/include*.

A log file describing the (main) changes from one JMVC software version to the next is given by *JMVC_changes.txt*. Note that this log files starts with the JMVC version 1.0 (CVS tag JMVC_1_0).

Table 2: Structure of the CVS repository for the JMVC software

Folder	***content***
JMVC	*source code and project files for the JMVC software* All files that are required for building and using the JMVC software are contained in this folder.
JMVC/H264Extension/build	*workspaces and makesfiles* Workspaces are provided for Microsoft Visual Studio 6 and Microsoft Visual Studio .NET. Makefiles are provided for Linux.
JMVC/H264Extension/data	*basic examples of encoder configuration files* These examples are not guaranteed to be complete or to contain all available configuration parameters. The examples only serve as a basis for writing encoder configuration files.
JMVC/H264Extension/include	*includes files that are required by other libraries inside the JMVC project* Only include files for classes that are required by other libraries or test projects should be placed into this folder. This folder contains a subfolder for the frequently used library *H264AVCCommonLib*. Include files of classes that are only required inside a library or test project are contained in the library or test project folder in *JMVC/H264Extension/src*.
JMVC/H264Extension/src	*include and source files for all libraries and test projects* This folder contains all source files and the include files that are only required inside a library or test project. This folder is organized in appropriate sub-folders.
Validation	*validation scripts for checking the JMVC software* Perl scripts for validating the JMVC software and is related to SVC and should be disregarded.
jmvc_changes.txt	*changes log file* This file described the (main) changes from one CVS version to the next. It starts with JMVC version 1.0 (CVS tag: JMVC_1_0).

1.3 Building the JMVC software

It shall be possible to build the JMVC software on a Windows32 platform with Microsoft Visual Studio .NET and on a Linux platform with gcc version 4. For information on how to build the software on a Windows32 platform with Microsoft Visual Studio .NET refer to section 1.3.1, and for information on how to build the software on a Linux platform with gcc version 4 refer to section 1.3.2.

Since the JMVC software is written in C++, it should also be possible to build the software on other platforms, which provide a C++ compiler. However, it is only guaranteed that the software can be build by using Microsoft Visual Studio .NET or the gcc compiler version 4.

All libraries are static libraries and all executable are statically linked to the libraries.

1.3.1 Windows32 platform with Microsoft Visual Studio

The folder *JMVC/H264Extension/build/windows* contains a Microsoft Visual Studio .NET workspace *H264AVCVideoEncDec.sln*. In order to build the software, open this workspace with Microsoft Visual Studio .NET, and build all project files by selecting *Build→Batch Build*, which opens a new dialog window. Then press the buttons *Select All* and *Rebuild*.

The folder *JMVC/H264Extension/build/windows* also contains a Microsoft Visual Studio 6 workspace *H264AVCVideoEncDec.dsw*. However, the compilation under Microsoft Visual Studio 6 is only occasionally checked, and thus it is not guaranteed that each software version can be build using Microsoft Visual Studio 6. In order to build the software with Microsoft Visual Studio 6, open the workspace with Microsoft Visual Studio 6.0, and build all project files by selecting

Build→Batch Build, which opens a new dialog window, in which the button *Rebuild All* shall be pressed.

After building the software the folders *bin* and *lib* shall contain the binaries and libraries summarized in Example 3. Note that there exist two different versions for each binary or library, one with and one without a "d" before the dot. The versions with a "d" before the dot represent binaries or libraries that have been built in debug mode, while the versions without a "d" before the dot represent binaries or libraries that have been built in release mode.

Example 3: Binaries and libraries after building the software on Windows

```
===== binaries =====
bin/DownConvertStatic.exe
bin/DownConvertStaticd.exe
bin/H264AVCDecoderLibTestStatic.exe
bin/H264AVCDecoderLibTestStaticd.exe
bin/H264AVCEncoderLibTestStatic.exe
bin/H264AVCEncoderLibTestStaticd.exe
bin/MVCBitStreamAssemblerStaticd.exe
bin/MVCBitStreamAssemblerStatic.exe
bin/MVCBitStreamExtractorStaticd.exe
bin/MVCBitStreamExtractorStatic.exe
bin/PSNRStatic.exe
bin/PSNRStaticd.exe


===== libraries =====
lib/H264AVCCommonLibStatic.lib
lib/H264AVCCommonLibStaticd.lib
lib/H264AVCDecoderLibStatic.lib
lib/H264AVCDecoderLibStaticd.lib
lib/H264AVCEncoderLibStatic.lib
lib/H264AVCEncoderLibStaticd.lib
lib/H264AVCVideoIoLibStatic.lib
lib/H264AVCVideoIoLibStaticd.lib
```

1.3.2 Linux platform with gcc compiler version 4

Makefiles for the Linux with gcc compiler are provided in the folder *JMVC/H264Extension/build/linux* and the corresponding sub-folders. Supposing that the current folder is the main folder *JMVC* of the JMVC repository (see section 1.2), the commands specified in Example 4 should be executed to build all project files.

Example 4: Building the JMVC software under Linux with a gcc compiler (version 4).

```
cd JMVC/H264AVCExtension/build/linux
make
```

By replacing *make* with *make release* or *make debug* in the Example 4 it can be specified that only the release or debug versions of the libraries and executables should be build.

After building the software the folders *bin* and *lib* shall contain the binaries and libraries summarized in Example 5. Note that there exist two different versions for each binary or library, one with and one without a "d" before the dot. The versions with a "d" before the dot represent binaries or libraries that have been built in debug mode, while the versions without a "d" before the dot represent binaries or libraries that have been built in release mode. When the command *make release* or *make debug* was used, only the debug or release version are present, respectively.

Example 5: Binaries and libraries after building the software on Linux

```
===== binaries =====
bin/DownConvertStatic
```

```
bin/DownConvertStaticd
bin/H264AVCDecoderLibTestStatic
bin/H264AVCDecoderLibTestStaticd
bin/H264AVCEncoderLibTestStatic
bin/H264AVCEncoderLibTestStaticd
bin/MVCBitStreamAssemblerStaticd
bin/MVCBitStreamAssemblerStatic
bin/MVCBitStreamExtractorStaticd
bin/MVCBitStreamExtractorStatic
bin/PSNRStatic
bin/PSNRStaticd


===== libraries =====
lib/libH264AVCCommonLibStatic.a
lib/libH264AVCCommonLibStaticd.a
lib/libH264AVCDecoderLibStatic.a
lib/libH264AVCDecoderLibStaticd.a
lib/libH264AVCEncoderLibStatic.a
lib/libH264AVCEncoderLibStaticd.a
lib/libH264AVCVideoIoLibStatic.a
lib/libH264AVCVideoIoLibStaticd.a
```

Information on binaries and libraries and Table 4 give information on the libraries and executables that are contained in the JMVC software package. Note that – as described in sections 1.3.1 and 1.3.2 – the naming of the actual library and executable files is dependent on the platform and on whether the libraries and executables have been built in release or debug mode.

Detailed information on command line options and configuration parameters for the executables are given in section 2. And in section 2.4 several examples for using the JMVC software are given in the form of a tutorial.

Table 3: Libraries provided by the JMVC software

library	***description***
H264AVCCommonLibStatic	*common lib* This library provides classes that are used by both the encoder and decoder, as for example macroblock data structures, buffers for storing and accessing image data, or algorithms for deblocking.
H264AVCEncoderLibStatic	*encoder lib* This library provides classes that are only used by the encoder. For example, it includes classes for motion estimation, mode decision, and entropy encoding.
H264AVCDecoderLibStatic	*decoder lib* This library provides classes that are only used by the decoder. For example, it includes classes for entropy decoding and bitstream parsing.
H264AVCVideoIoLibStatic	*io lib* This library provides classes for reading and writing NAL units in the byte-stream format as well as classes for reading and writing raw video data.

Table 4: Executables provided by the JMVC software

executable	***description***
DownConvertStatic	*resampler* The resampler can be used for spatial/temporal resampling (up-sampling or down-sampling) of video sequences. More information on using the resampler are provided in section Error: Reference source not found.
H264AVCEncoderLibTestStatic	*AVC/MVC encoder* The encoder can be used for generating single-view (MVC) or

	multivew (MVC) bitstreams. More information on using the encoder is provided in section 2.1.
H264AVCDecoderLibTestStatic	*MVC decoder* The decoder can be used for decoding AVC or MVC bitstreams and reconstructing raw video sequences. More information on using the decoder are provided in section 2.2
MVCBitStreamAssembler	*MVC assembler* The bitstream assembler should be executed after running the encoder to generate a bitstream containing all the encoded views. The assembling of the views is done in time-first order. More information on using the assembler are provided in section 2.4
MVCBitStreamExtractor	*MVC extractor* The bitstream extractor can be used to extract sub-streams of an MVC stream. The substreams represent streams with a reduced num of views. More information on using the decoder are provided in section 2.5
PSNRStatic	*PSNR tool* The PSNR tool can be used for measuring the PSNR between two raw video sequences. In addition it can be used for measuring the bit-rate of a given bitstream. More information on using the PSNR tool are provided in section 2.3.

2 Usage and configuration of the JMVC software

This section provides information on usage and configuration of the binaries contained in the JMVC software package.

2.1 Encoder "H264AVCEncoderLibTestStatic"

The encoder can be used for generating AVC or MVC bitstreams. The basic encoder call is illustrated in Example 6. At this *mcfg* represents the filename of the main configuration file. The main configuration file shall be specified for each encoder call. *view_id* represents the view that should be coded. The encoder should be run for each view that is to be encoded.

Example 6: Using the encoder

```
H264AVCEncoderLibTestStatic.exe –vf <mcfg> <view_id>
```

It should be noted that using the encoder does not guarantee rate-distortion efficient coding. For obtaining optimized encoding results the encoder configuration has to be carefully specified. It should further be noted that the current encoder implementation does not provide a rate-control. The bit-rate needs to be controlled by selecting appropriate quantization parameters. Examples for using the encoder are described in section 3.2.

Generally, the configuration files present a collection of configuration parameters. Each configuration parameter is specified in one line of the configuration files. Comments are started by the character '#'. The order of configuration parameters inside a configuration file can be arbitrarily selected. Each configuration parameter has a default value, and when the configuration parameter is not present in the configuration file, the default value is taken instead. Thus, it is generally not required to specify all configuration parameters in a configuration file.

2.1.1.1 Configuration file

All available configuration file parameters for the multiview mode together with a brief description are summarized in Example 7. Additional information about the configuration parameters, including default values, is given below.

The current JMVC software can generate bitstreams from a configuration file that has a large GOP size and a large number of views, however such bitstreams may not be conforming and are for research purposes.

Example 7: Encoder configuration file in multiview coding mode

```
# JMVC Configuration File in MVC mode

#====================== GENERAL =================================================
InputFile              input      # input file
OutputFile             stream     # bitstream file
ReconFile              rec        # reconstructed file
MotionFile             motion     # motion information file
SourceWidth            640        # input  frame width
SourceHeight           480        # input  frame height
FrameRate              25.0       # frame rate [Hz]
FramesToBeEncoded      250        # number of frames

#====================== CODING ==================================================
SymbolMode             1          # 0=CAVLC, 1=CABAC
FRExt                  1          # 8x8 transform (0:off, 1:on)
BasisQP                31         # Quantization parameters

#====================== INTERLACED ======================================
MbAff                  0          # 0=frameMb, 1=MbAff
PAff                   0          # 0=frame, 1=field, 2=frame/field

#====================== STRUCTURE ==============================================
GOPSize                12          # GOP Size (at maximum frame rate)
IntraPeriod            12         # Anchor Period
NumberReferenceFrames  2          # Number of reference pictures
InterPredPicsFirst     1          # 1 Inter Pics; 0 Inter-view Pics
DeltaLayer0Quant       0          # differential QP for layer 0
DeltaLayer1Quant       3          # differential QP for layer 1
DeltaLayer2Quant       4          # differential QP for layer 2
DeltaLayer3Quant       5          # differential QP for layer 3
DeltaLayer4Quant       6          # differential QP for layer 4
DeltaLayer5Quant       7          # differential QP for layer 5
PicOrderCntType        0          # Picture order count type (0 or 2)

#=============================== MOTION SEARCH ==================================
SearchMode             4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchFuncFullPel      3          # Search function full pel
                                  #   (0:SAD, 1:SSE, 2:HADAMARD, 3:SAD-YUV)
SearchFuncSubPel       2          # Search function sub pel
                                  #   (0:SAD, 1:SSE, 2:HADAMARD)
SearchRange            32         # Search range (Full Pel)
BiPredIter             4          # Max iterations for bi-pred search
IterSearchRange        8          # Search range for iterations (0: normal)

#=============================== LOOP FILTER ====================================
LoopFilterDisable      0          # Loop filter idc (0: on, 1: off, 2:
                                  #   on except for slice boundaries)
LoopFilterAlphaC0Offset 0         # AlphaOffset(-6..+6): valid range
LoopFilterBetaOffset   0          # BetaOffset (-6..+6): valid range

#=============================== WEIGHTED PREDICTION ============================
WeightedPrediction     0          # Weighting IP Slice (0:disable, 1:enable)
WeightedBiprediction   0          # Weighting B  Slice (0:disable, 1:explicit,
                                  #                      2:implicit)

#=============================== NESTING SEI MESSAGE =============================
NestingSEI             0          #(0: NestingSEI off, 1: NestingSEI on)
SnapShot               0          #(0: SnapShot off, 1: SnapShot on)
#=========================== ACTIVE VIEW INFO SEI MESSAGE ========================
ActiveViewSEI          0          #(0: ActiveViewSEI off, 1: ActiveViewSEI on)
#===================== VIEW SCALABILITY INFOMATION SEI MESSAGE ==================
ViewScalInfoSEI        0          #(0: ViewScalSEI off, 1: ViewScalSEI on)
```

```
#===================== MULTIVIEW SCENE INFORMATION SEI MESSAGE ==================
MultiviewSceneInfoSEI           1 #(0: off, 1: on)
MaxDisparity              12
#=================MULTIVIEW ACQUISITION INFOMATION SEI MESSAGE ================
MultiviewAcquisitionInfoSEI     1 #(0: off, 1: on)
AcquisitionInfoFile       Camera_ballroom.cfg


#=================== PARALLEL DECODING INFORMATION SEI Message ==================
PDISEIMessage           0          # PDI SEI message enable (0: disable, 1:enable)
PDIInitialDelayAnc      2          # PDI initial delay for anchor pictures
PDIInitialDelayNonAnc   2          # PDI initial delay for non-anchor pictures

#============== Level conformance checking of the DPB size ==============
DPBConformanceCheck      1      # (0: disable, 1: enable, 1:default)

NumViewsMinusOne        2          # (Number of view to be coded minus 1)
ViewOrder               0-2-1      # (Order in which view_ids are coded)

View_ID             0          # (view_id of a view 0 - 1024)
Fwd_NumAnchorRefs   0          # (number of list_0 references for anchor)
Bwd_NumAnchorRefs   0          # (number of list 1 references for anchor)
Fwd_NumNonAnchorRefs    0          # (number of list 1 references for non-anchor)
Bwd_NumNonAnchorRefs    0          # (number of list 1 references for non-anchor)

View_ID              1
Fwd_NumAnchorRefs    1
Bwd_NumAnchorRefs    1
Fwd_NumNonAnchorRefs     1
Bwd_NumNonAnchorRefs     1
Fwd_AnchorRefs       0 0
Bwd_AnchorRefs        0 2
Fwd_NonAnchorRefs    0 0
Bwd_NonAnchorRefs    0 2

View_ID             2
Fwd_NumAnchorRefs    1
Bwd_NumAnchorRefs    0
Fwd_NumNonAnchorRefs     0
Bwd_NumNonAnchorRefs     0
Fwd_AnchorRefs         0 0


NumLevelValuesSignalledMinus1 0

Level_IDC 1
NumApplicableOpsMinus1  0
ApplicableOpTemporalId 0 0
ApplicableOpNumTargetViewsMinus1 0 2
ApplicableOpNumViewsMinus1 0 2
ApplicableOpTargetViewId 0 0 0
ApplicableOpTargetViewId 0 1 2
ApplicableOpTargetViewId 0 2 1
```

InputFile

String, default: "in"

Specifies the filename (without the .yuv) of the original raw video sequence to be encoded. The input files should have the format in_0.yuv, in_1.yuv etc for the different set of views.

OutputFile

String, default: "test"

Specifies the filename (without the .264) of the bitstream to be generated. *test_0.264, test_1.264 etc* will be created for the different views automatically.

ReconFile

String, default: "rec"

Specifies the filename (without the .yuv) of the coded and reconstructed input sequence. This sequence is provided for debugging purposes. *rec_0.yuv, rec_1.yuv etc*. will automatically be created by the encoder.

MotionFile

String, default: "motion"

Specifies the filename (without the .dat) of the sequence to be generated. This sequence is provided for motion skip mode. *motion_0.dat, motion_1.dat etc*. will automatically be created by the encoder.

SourceWidth

Unsigned Int, default: 0 (invalid)

Specifies the width of the input images in luma samples. *SourceWidth* shall be non-zero and a multiple of 16. This parameter shall be present in each configuration file, since the default value of 0 is invalid.

SourceHeight

Unsigned Int, default: 0 (invalid)

Specifies the height of the input images in luma samples. *SourceHeight* shall be non-zero and a multiple of 16. This parameter shall be present in each configuration file, since the default value of 0 is invalid.

FrameRate

Double, default: 60.0

Specifies the frame rate of the input sequence in Hz.

FramesToBeEncoded

Unsigned Int, default: 1

Specifies the number of frames of the input sequence to be encoded for one view.

SymbolMode

Flag (0 or 1), default: 1

Specifies the entropy coding mode. When *SymbolMode* is equal to 0, the video sequence is encoded using variable length codes (VLC). When *SymbolMode* is equal to 1, the video sequence is encoded using context-adaptive binary arithmetic coding (CABAC). CABAC usually provides an increased coding efficiency.

FRExt

Flag (0 or 1), default: 1

Specifies whether the 8x8 transform (High Profile) is enabled. When *FRExt* is equal to 1, the 8x8 transform enabled in addition to the standard 4x4 transform for the luminance component; otherwise the 8x8 transform is disabled. The coding efficiency is generally increased by enabling the 8x8 transform, especially for high-resolution source material.

BasisQP

Double, default: 26

Specifies the basic quantization parameter. This parameter shall be used to control the bit-rate of a bitstream. The actual quantization parameters that are used for encoding a specific frame of a video sequence are additionally dependent on the parameter *SequenceFormatString* and the parameters *DeltaLayerXQuant* (with X in the range of 0 to 5, inclusive). More information on how quantization parameters for specific frames are chosen is given in the description of the parameters *DeltaLayerXQuant*.

MbAff

Flag (0 or 1), default: 0

Specifies whether Macroblock Adaptive Frame/Field Coding is enabled. When *PAff* is not equal to 1 and *MbAff* is equal to 1, each macroblock is adaptively selected to be frame coded or field coded. *MbAff* is equal to 0, each macroblock is frame coded.

PAff

Unsigned Int, default: 0

Specifies the picture coding structure used. *PAff can be 0, 1 or 2.* When *PAff* is equal to 0, the picture coding structure used is frame picture. When *PAff* is equal to 1, the picture coding structure used is field picture. And when *PAff* is equal to 2, the picture coding structure used is adaptively selected as frame or field picture Setting *PAff to 1 or 2*, may be preferable for the coding of interlaced scan source material.

`GOPSize`

Unsigned Int, default: 1

Specifies the GOP size that shall be used for encoding a video sequence. A GOP (group of pictures) consists of an anchor picture and several hierarchically coded B pictures that are located between the anchor pictures. The parameter *GOPSize* must be equal to a power of 2 or 12 and 15 for MVC. The GOP size is specified at the frame rate given by *FrameRate*. The maximum allowed value is 64.

`NumReferenceFrames`

Specifies the maximum number of active entries for the temporal reference pictures lists 0 and 1. The actual number of active entries that are used for encoding a specific frame, is additionally dependent on the location of a frame inside the group of pictures.

`InterPredPicsFirst`

Specifies how encoder reorders the reference picture lists. InterPredPicsFirst equal to 1 means inter prediction reference pictures are in front of the inter-view prediction pictures in the reference picture lists, it equal to 0 means inter prediction pictures are behind the inter-view prediction pictures in the reference picture lists.

`DeltaLayerXQuant` (with X being replaced by 0, 1, 2, 3, 4, or 5)

Double, default: 0

Specifies the quantization offset for the frames of layer X. The quantization parameter of a frame of layer X is calculated by *QP = BasisQP + DeltaLayerXQuant*.

`PicOrderCntType`

Unsigned Int, default: 0

Specifies the PicOrderCnt type of the coded video sequence. PicOrderCntType can be 0 or 2. When the value is 2, *GOPSize* must be 1.

`SearchMode`

Int, default: 0

Specifies the motion search algorithm to be applied. When *SearchMode* is equal to 0, an exhaustive block search is employed. When *SearchMode* is equal to 4, a fast motion search algorithm is employed. Other values than 0 or 4 lead to an unspecified behaviour. The fast motion search algorithm shall be preferred, since it provides a comparable rate-distortion efficiency, but significantly reduced the encoding time.

`SearchFuncFullPel`

Int, default: 0

Specifies the distortion measure that is applied for the motion search on integer-sample positions. The following values are supported:

0 – Sum of absolute differences (SAD) for the luminance component
1 – Sum of squared differences (SSE) for the luminance component
2 – Sum of absolute differences in the Hadamard transform domain for the luminance component
3 – Sum of absolute differences (SAD) for all colour components

`SearchFuncSubPel`

Int, default: 0

Specifies the distortion measure that is applied for the motion search on sub-sample positions. The following values are supported:

0 – Sum of absolute differences (SAD) for the luminance component
1 – Sum of squared differences (SSE) for the luminance component
2 – Sum of absolute differences in the Hadamard transform domain for the luminance component

`SearchRange`

Unsigned Int, default: 96

Specifies the maximum search range for the motion search. Note that when the fast search algorithm is selected, the actual search range can be smaller.

`BiPredIter`

Unsigned Int, default: 4

Specifies the number of iterations of the motion search for bi-predictive blocks. The coding efficiency for B pictures is usually increased when the parameter *BiPredIter* is set to a value greater than or equal to 2.

`IterSearchRange`

Unsigned Int, default: 8

Specifies the search range for the motion search iterations for bi-predictive blocks. Since, an initial search is performed with the search range specified by *SearchRange*, this parameter can be set to significantly smaller values without decreasing the coding efficiency.

`LoopFilterDisable`

Unsigned Int, default: 0

Specifies how the in-loop deblocking filter is applies. The following values are supported:

0 – The deblocking filter is applied to all block edges

1 – The deblocking filter is not applied.

2 – The deblocking filter is applied to all block edges with exception of slice boundaries

`LoopFilterAlphaC0Offset`

Int, default: 0

Specifies the alpha offset for the deblocking filter. *LoopFilterAlphaC0Offset* shall be in the range of –6 to 6, inclusive. This parameter can be used to adjust the strength of the deblocking filter.

`LoopFilterBetaOffset`

Int, default: 0

Specifies the beta offset for the deblocking filter. *LoopFilterBetaOffset* shall be in the range of –6 to 6, inclusive. This parameter can be used to adjust the strength of the deblocking filter.

`WeightedPrediction`

Flag (0 or 1), default: 0

Specifies whether weighted prediction is used for P pictures. When this parameter is equal to 0, weighted prediction for P pictures is disabled. When this parameter is equal to 1, weighted prediction for P pictures is enabled, and the prediction weights are estimated during encoding.

`WeightedBiPrediction`

Unsigned Int, default: 0

Specifies whether and how weighted prediction is used for B pictures. When this parameter is equal to 0, weighted prediction for B pictures is disabled. When this parameter is equal to 1, weighted prediction for B pictures is enabled and operated in explicit weighting mode; the corresponding prediction weights are estimated during encoding. When this parameter is equal to 2, weighted prediction for B pictures is enabled and operated in implicit prediction mode. In the implicit mode, the prediction weights are not estimated and transmitted, but inferred from the distances (measured via Picture Order Count) of the reference pictures to the picture currently to be encoded.

`NestingSEI`

Unsigned Int, default: 0

Specifies whether Scalable nesting SEI message is used or not. When this parameter is equal to 0, NestingSEI is not used. When this parameter and parameter SnapShot is equal to 1, NestingSEI is used.

`SnapShot`

Unsigned Int, default: 0

Specifies whether full-frame snapshot SEI message is used or not. When this parameter is equal to 0, snapshot SEI is not used. When this parameter is equal to 1, snapshot SEI is used.

`ActiveViewSEI`

Unsigned Int, default: 0

Specifies whether Active View Information SEI (AVI) message is used or not. When this parameter is equal to 0, AVI is not used. When this parameter is equal to 1, AVI is used.

`ViewScalInfoSEI`

Unsigned Int, default: 0

Specifies whether View scalability information SEI (ViewScalInfo) message is used or not. When this parameter is equal to 0, ViewScalInfo is not used. When this parameter is equal to 1, ViewScalInfo is used.

`MultiviewSceneInfoSEI`

Unsigned Int, default: 0

Specifies whether Multiview Scene Information SEI (MultiviewSceneInfo) message is used or not. When this parameter is equal to 0, MultiviewSceneInfo is not used. When this parameter is equal to 1, MultiviewSceneInfo is used.

`MaxDisparity`

Unsigned Int, default:0

Specifies the maximum disparity only when MultiviewSceneInfoSEI is set to 1. The value should be greater than or equal to zero if valid.

`MultiviewAcquisitionInfoSEI`

Unsigned Int, default: 0

Specifies whether Multiview Acquisition Information SEI (MultiviewAcquisitionInfo) message is used or not. When this parameter is equal to 0, MultiviewAcquisitionInfo is not used. When this parameter is equal to 1, MultiviewAcquisitionInfo is used.

`AcquisitionInfoFile`

Specifies the filename which contains the multiview acquisition information only when MultiviewAcquisitonInfoSEI is set to 1.

* The format of the AcquisitionInfoFile for Multiview Acquisition SEI message is illustrated in example 12-1 below (i.e. Camera_ballroom.cfg in example 12 should be as follows). Detailed description of the parameters can be found in JVT-Z038 and JVT-W060.

Example 12-1: Configuration file for multiview acquisition information

```
NumViewsMinus1 1 # number of views - 1
IntrinsicParameterFlag  1    # intrinsic_param_flag
IntrinsicParametersEqual 0    # intrinsic_params_equal
ExtrinsicParameterFlag 1    # extrinsic_param_flag
Precision_FocalLength 8
Precision_PrincipalPoint 8
Precision_RadialDistortion 8
Precision_RotationParam 9
Precision_TranslationParam 10

View_ID  0
FocalLengthX 5    # focal_length_x (unit: pixels)
FocalLengthY 6    # focal_length_y (unit: pixels)

PrincipalPointX 4   # principal_point_x (unit: pixels)
PrincipalPointY 3   # principal_point_y (unit: pixels)
RadialDistortion 2   # radial_distortion
R_1 10 11 23    # r_11 r_12 r_13
R_2 32 33 22    # r_21 r_22 r_23
R_3 10  1  3    # r_31 r_32 r_33
Translation  1 5 4    # t_1  t_2  t_3

View_ID  1
FocalLengthX 51
FocalLengthY 61
PrincipalPointX 41
PrincipalPointY 31
RadialDistortion 21
R_1 101 111 231
R_2 321 331 221
R_3 10 11  13

Translation  11 51 41
```

PDISEIMessage

Unsigned Int, default: 0

Specifies whether Parallel Decoding Information (PDI) SEI message is used or not. When this parameter is equal to 0, PDI is not used. When this parameter is equal to 1, PDI is used.

PDIInitialDelayAnc

Unsigned Int, default:0

Specifies the available reference area for anchor pictures only when PDISEIMessage is set to 1. The value should be greater than or equal to 2 if valid.

PDIInitialDelayNonAnc

Unsigned Int, default:0

Specifies the available reference area for non-anchor pictures only when PDISEIMessage is set to 1. The value should be greater than or equal to 2 if valid.

DPBConformanceCheck

Unsigned Int, default:1

Specifies whether to enable/disable(1/0) the level conformance checking of the DPB size. When it is not present in the cfg file, it is assumed to be 1 by default.

NumViewsMinusOne

Unsigned Int (0-1023): 7

Specifies the number of views to be encoded minus 1. This parameter can contain a value between 0 and 1023 which is the maximum number of views currently allowed.

ViewOrder

String (numbers): 0

Specifies the order in which the views are to be coded. This numbers signify the view_id of the views.

View_ID

Unsigned Int (0-1024): 0

Specified the view identifier (view_id) of the view to be encoded and described by the parameters that follow.

Fwd_NumAnchorRefs

Unsigned Int (0-xx): 0

Specifies the number of list 0 references to be used for the anchor pictures of the view identified by View_ID.

Bwd_NumAnchorRefs

Unsigned Int (0-xx): 0

Specifies the number of list 1 references to be used for the anchor pictures of the view identified by View_ID.

Fwd_NumNonAnchorRefs

Unsigned Int (0-xx): 0

Specifies the number of list 0 references to be used for the non-anchor pictures of the view identified by View_ID.

Bwd_NumNonAnchorRefs

Unsigned Int (0-xx): 0

Specifies the number of list 1 references to be used for the non-anchor pictures of the view identified by View_ID.

Fwd_AnchorRefs

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the relative refrence index position for the list 0 reference for anchor pictures and the second number indicates reference view_id. **Currently only one value is allowed for the relative reference index position and it should be set to 0.**

Bwd_AnchorRefs

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the relative refrence index position for the list 1 reference for anchor pictures and the second number indicates reference view_id. **Currently only one value is allowed for the relative reference index position and it should be set to 0.**

`Fwd_NonAnchorRefs`

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the relative refrence index position for the list 0 reference for non-anchor pictures and the second number indicates reference view_id. **Currently only one value is allowed for the relative reference index position and it should be set to 0.**

`Bwd_NonAnchorRefs`

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the relative refrence index position for the list 0 reference for non-anchor pictures and the second number indicates reference view_id. **Currently only one value is allowed for the relative reference index position and it should be set to 0.**

`NumLevelValuesSignalledMinus1`

Unsigned Int (0-63): 0

Specfies the number of level values signalled minus 1 for the coded video sequence. The value shall be in the range of 0 to 63, inclusive.

`Level_IDC`

Unsigned Int (0-1023): 0

Specifies the level identifier (level_idc) and described by the parameters that follow.

`NumApplicableOpsMinus1`

Unsigned Int (0-1023): 0

plus 1 specifies the number of operation points to which the level indicated by Level_IDC that precedes applies. The value shall be in the range of 0 to 1023, inclusive.

`ApplicableOpTemporalId`

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the index of the operation point to which the level indicated by Level_IDC that precedes applies. The second number indicates the temporal_id of the corresponding operation point and its value shall be in the range of 0 to 1023, inclusive.

`ApplicableOpNumTargetViewsMinus1`

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the index of the operation point to which the level indicated by Level_IDC that precedes applies. The second number indicates the number of target output views minus 1 for the corresponding operation point and its value shall be in the range of 0 to 1023, inclusive.

`ApplicableOpNumViewsMinus1`

Pair of Unsigned Int:

Specifies a pair of values. The first number indicates the index of the operation point to which the level indicated by Level_IDC that precedes applies. The second number indicates the number of views, including the views that are dependent on the target output views but do not belong to the target output views for the corresponding operation point and its value shall be in the range of 0 to 1023, inclusive.

`ApplicableOpTargetViewId`

Triplelet of Unsigned Int:

The first number indicates the index of the operation point to which the level indicated by Level_IDC that precedes applies. The second number indicates the index of the target view. The third number specifies the target output view (applicable_op_target_view_id**)** for the corresponding operation point and its value shall be in the range of 0 to 1023, inclusive.

2.2 Decoder “H264AVCDecoderLibTestStatic”

The decoder call is illustrated in Example 8. The parameter *str* specifies the filename of the bitstream to be decoded, and the parameter *rec* specifies the filename for the reconstructed video sequence.

Example 8: Using the decoder

```
H264AVCDecoderLibTestStatic <str> <rec> <numViews>

       str: bitstream file (input)
       rec: reconstructed video sequence (output)
       numViews: number of view in bitstream
```

2.3 PSNR tool “PSNRStatic”

The PSNR tool can be used for measuring the Peak-Signal-To-Noise-Ratio (PSNR) between two sequences, it can additionally be used for calculating the bit-rate. The usage of the PSNR tool is illustrated in Example 9. The PSNR of every frame is written to *stdout*, and the average PSNR over all frames is written to *stderr*.

Example 9: Using the PSNR tool

```
PSNRStatic <w> <h> <org> <rec> [<t> [<skip> [<strm> <fps>]]]

    w: original width  (luma samples)
    h: original height (luma samples)
  org: original file
  rec: reconstructed file
    t: number of temporal downsampling stages (default: 0)
 skip: number of frames to skip at start      (default: 0)
 strm: coded stream
 frms: frames per second
```

The filenames for the original sequence and the reconstructed sequence are specified by the parameters *org* and *rec*. The spatial resolution of the original sequence is specified by the frame width *w* and the frame height *h*. By default, the temporal resolution of the reconstructed sequence shall be identical to the temporal resolution of the original sequence. It is however also possible to measure the PSNR between the original and reconstructed sequence, when the reconstructed sequence represents a temporally downsampled version. In that case, the temporal downsampling stages *t* need to be specified. This parameter is identical to the one described in section Error: Reference source not found.

The optional parameter *skip* can be used to specify that *skip* frames at the start of the sequences shall be skipped. The bit-rate of a corresponding bitstream is additionally outputted when a bitstream file *strm* and a frame rate (in Hz) *fps* is specified. It is not possible to specify only one of these parameters.

Example 10: Using the PSNR tool for PSNR and rate measurement

```
PSNRStatic 176 144 org.yuv rec.yuv 0 0 str.svc 15 2>PSNR.txt

type PSNR.txt
128,00  32,23  38,79  39.02
```

In Example 10, the most common use of the PSNR tool is illustrated. The files “org.yuv” and “rec.yuv” specify the original and reconstructed sequences, respectively. The file “str.svc” specifies a bitstream. It is assumed that the sequences are given in QCIF resolution (176x144 samples) and a frame rate of 15 Hz. The PSNR is measured between the original and reconstructed sequence, and the bit-rate of the given stream is calculated. The average bit-rate as well as the PSNR values for the luminance and the two chrominance components are written to the file “PSNR.txt”. All values are written to one line of the file and are separated by tabulator characters. The order of the values is the following: (1) bit-rate in kbit/s, (2) Y-PSNR in dB – luminance component, (3) U-PSNR in dB – chrominance component U or Cb, (4) V-PSNR in dB – chrominance component V or Cr.

2.4 Assembler Tool for MVC ”MVCBitStreamAssembler”

The MVCBitStreamAssembler tool is used to assemble the individual bitstreams that represent the different views into a single bitstream before the decoding begins. A configuration file is used to specify in what order the bitstreams need to be assembler. This order should be the view coding order. The assembler assembles the bitstream in the view coding order such that the pictures at a certain time instance are together. This represents the time-first coding. The configuration file for the assembler is shown below.

Example 11: Example assembler configuration file “assembler.cfg”

```
#====================== Assembler: View Encode order =========================
OutputFile              ballroom.264
NumberOfViews           8
InputFile0              stream_0.264
InputFile1              stream_2.264
InputFile2              stream_1.264
InputFile3              stream_4.264
InputFile4              stream_3.264
InputFile5              stream_6.264
InputFile6              stream_5.264
InputFile7              stream_7.264
```

An example of using the assembler is shown below.

Example 12: Assembler example.

```
>MVCBitStreamAssemblerStatic –vf assembler.cfg
```

2.5 BitStream Extractor for MVC ”MVCBitStreamExtractorStatic”

The bitstream extractor can be used to extract sub-streams of an MVC stream. The substreams represent streams with a reduced num of views. The usage of the bitstream extractor is illustrated below. A printout of the options that are provided by the bitstream extractor can be obtained by calling the extractor without any command lin parameter.

Example 13: Using the MVC bitstream extractor

```
MVCBitStreamExtractorStaticd InputStream [OutputStream | [-op] ]
options:
        -op Op -> extract the corresponding packets with operation point id = Op
```

The parameter *InputStream* specifies the filename for the input bitstream (global bitstream). The parameter *OutputStream* specifies the filename for the output bitstream, which generally represents a sub-stream of the input bitstream as specified by the additional command line parameters.

When the extractor is called with a single parameter *input.264* specifying the input bitstream, information about the containe view scalability information SEI message are displayed as illustrated below.

Example 14: Using the MVC bitstream extractor for displaying information about a bitstream

```
>MVCBitStreamExtractorStatic input.264

JMVC 4.0 BitStream Extractor
```

```
Total number of operation points: 8

NO.0 of opertion points:
Operation Point Id: 0
Priority Id: 4
Temporal Id: 4
Number of active views: 1
View_Id[0]: 0
No profile and level information.
Avg Bitrate: 11
Max Bitrate: 20
Max Bitrate Calc Window: 100
Constant Frame rate Idc: 0
Avg Frame rate: 25
No OpDependency information.
No SPS/PPS required information.


NO.1 of opertion points:
Operation Point Id: 1
Priority Id: 5
Temporal Id: 4
Number of active views: 1
View_Id[0]: 2
No profile and level information.
Avg Bitrate: 12
Max Bitrate: 22
Max Bitrate Calc Window: 100
Constant Frame rate Idc: 0
Avg Frame rate: 25
Number of Directly Dependent Operation points: 1
NO1 of the dependent operation points(operation point id): 0
No SPS/PPS required information.

...
```

The option *–op* is used to specify the operation point id that should be extracted. The output is all the packets that will be used in the decoding of the specified operation point id. An example of using this is shown below.

Example 15: Extraction of a specified operation point id

```
>MVCBitStreamExtractorStatic input.264 output.264 -op 0
```

3 Use Examples as a brief tutorial

In the following subsection, the usage of the JMVC software is illustrated by means of examples for frequently used scenarios.

3.1 Original sequences generation

This section aims at providing the detailed process on how to generate the rectified MVC sequences. These original sequences can be downloaded via the information described in JVT-T207.

3.2 Multiview coding Example

We look at multiview coding with the JMVC software. Let's assume, we want to encode a given sequence "ballroom" with VGA (640x480) resolution, a frame rate of 25 Hz and with 8 views. Each file should then be named in a format similar to "ballroom_640x480_25p_0.yuv" for view 0, "ballroom_640x480_25p_1.yuv" for view 1 and so on. Example for configuration file is depicted in Example 16.

Example 16: Example main configuration file "encoderMVC.cfg"

```
# JMVC Main Configuration File

#======================= GENERAL =========================================
InputFile               input      # input file
OutputFile              stream     # bitstream file
ReconFile               rec        # reconstructed file
MotionFile              motion     # motion information file
SourceWidth             640        # input  frame width
SourceHeight            480        # input  frame height
FrameRate               25.0       # frame rate [Hz]
FramesToBeEncoded       250        # number of frames

#======================= CODING ==========================================
SymbolMode              1          # 0=CAVLC, 1=CABAC
FRExt                   1          # 8x8 transform (0:off, 1:on)
BasisQP                 31         # Quantization parameters

#======================= INTERLACED ======================================
MbAff                   0          # 0=frameMb, 1=MbAff
PAff                    0          # 0=frame, 1=field, 2=frame/field

#======================= STRUCTURE =======================================
GOPSize                 4           # GOP Size (at maximum frame rate)
IntraPeriod             12         # Anchor Period
NumberReferenceFrames   3          # Number of reference pictures
InterPredPicsFirst      1          # 1 Inter Pics; 0 Inter-view Pics
Log2MaxFrameNum         11         # specifies max. value for frame_num (4..16)
Log2MaxPocLsb           7          # specifies coding of POC's (4..15)
DeltaLayer0Quant        0          # differential QP for layer 0
DeltaLayer1Quant        3          # differential QP for layer 1
DeltaLayer2Quant        4          # differential QP for layer 2
DeltaLayer3Quant        5          # differential QP for layer 3
DeltaLayer4Quant        6          # differential QP for layer 4
DeltaLayer5Quant        7          # differential QP for layer 5
MaxRefIdxActiveBL0      2          # active entries in ref list 0 for B slices
MaxRefIdxActiveBL1      2          # active entries in ref list 1 for B slices
MaxRefIdxActiveP        1          # active entries in ref list for P slices

#======================== MOTION SEARCH ==================================
SearchMode              4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchFuncFullPel       3          # Search function full pel
                                   #   (0:SAD, 1:SSE, 2:HADAMARD, 3:SAD-YUV)
SearchFuncSubPel        2          # Search function sub pel
                                   #   (0:SAD, 1:SSE, 2:HADAMARD)
SearchRange             32         # Search range (Full Pel)
BiPredIter              4          # Max iterations for bi-pred search
IterSearchRange         8          # Search range for iterations (0: normal)

#========================= LOOP FILTER ===================================
LoopFilterDisable       0          # Loop filter idc (0: on, 1: off, 2:
                                   #  on except for slice boundaries)
LoopFilterAlphaC0Offset 0          # AlphaOffset(-6..+6): valid range
LoopFilterBetaOffset    0          # BetaOffset (-6..+6): valid range

#========================== WEIGHTED PREDICTION ===========================
WeightedPrediction      0          # Weighting IP Slice (0:disable, 1:enable)
WeightedBiprediction    0          # Weighting B  Slice (0:disable, 1:explicit,
                                                         2:implicit)
#=================== PARALLEL DECODING INFORMATION SEI Message ==================
PDISEIMessage           0          # PDI SEI message enable (0: disable, 1:enable)
PDIInitialDelayAnc      2          # PDI initial delay for anchor pictures
PDIInitialDelayNonAnc   2          # PDI initial delay for non-anchor pictures


#=============================== NESTING SEI MESSAGE =============================
NestingSEI              0          #(0: NestingSEI off, 1: NestingSEI on)
SnapShot                0          #(0: SnapShot off, 1: SnapShot on)
#========================== ACTIVE VIEW INFO SEI MESSAGE ========================
ActiveViewSEI           0          #(0: ActiveViewSEI off, 1: ActiveViewSEI on)
#===================== VIEW SCALABILITY INFOMATION SEI MESSAGE ==================
ViewScalInfoSEI         0          #(0: ViewScalSEI off, 1: ViewScalSEI on)

#============== Level conformance checking of the DPB size ==============
DPBConformanceCheck      1      # (0: disable, 1: enable, 1:default)

#================= MULTIVIEW CODING PARAMETERS ===========================
NumViewsMinusOne        7          # (Number of view to be coded minus 1)
ViewOrder               0-2-1-4-3-6-5-7   # (Order in which view_ids are coded)

View_ID                 0          # view_id (0..1024): valid range
Fwd_NumAnchorRefs       0          # (number of list_0 references for anchor)
Bwd_NumAnchorRefs        0      # (number of list 1 references for anchor)
Fwd_NumNonAnchorRefs     0      # (number of list 0 references for non-anchor)
Bwd_NumNonAnchorRefs     0      # (number of list 1 references for non-anchor)

View_ID                  1          # view_id (0..1024): valid range
Fwd_NumAnchorRefs        1          # (number of list_0 references for anchor)
Bwd_NumAnchorRefs        1          # (number of list 1 references for anchor)
Fwd_NumNonAnchorRefs      1         #(number of list 0 references for non-anchor)
Bwd_NumNonAnchorRefs     1         #(number of list 1 references for non-anchor)
Fwd_AnchorRefs           0 0      #ref_idx view_id combination
Bwd_AnchorRefs           0 2      #ref_idx view_id combination
Fwd_NonAnchorRefs        0 0      #ref_idx view_id combination
Bwd_NonAnchorRefs        0 2      #ref_idx view_id combination

View_ID                  2
Fwd_NumAnchorRefs        1
Bwd_NumAnchorRefs        0
Fwd_NumNonAnchorRefs     0
Bwd_NumNonAnchorRefs     0
Fwd_AnchorRefs           0 0

View_ID                  3
Fwd_NumAnchorRefs        1
Bwd_NumAnchorRefs        1
Fwd_NumNonAnchorRefs     1
Bwd_NumNonAnchorRefs     1
Fwd_AnchorRefs           0 2
Bwd_AnchorRefs           0 4
Fwd_NonAnchorRefs        0 2
Bwd_NonAnchorRefs        0 4

View_ID                 4
Fwd_NumAnchorRefs        1
Bwd_NumAnchorRefs        0
Fwd_NumNonAnchorRefs     0
Bwd_NumNonAnchorRefs     0
Fwd_AnchorRefs           0 2

View_ID                 5
Fwd_NumAnchorRefs        1
Bwd_NumAnchorRefs        1
Fwd_NumNonAnchorRefs     1
Bwd_NumNonAnchorRefs     1
Fwd_AnchorRefs           0 4
Bwd_AnchorRefs           0 6
Fwd_NonAnchorRefs        0 4
Bwd_NonAnchorRefs        0 6

View_ID                 6
Fwd_NumAnchorRefs        1
Bwd_NumAnchorRefs        0
Fwd_NumNonAnchorRefs     0
Bwd_NumNonAnchorRefs     0
Fwd_AnchorRefs            0 4

View_ID                 7
```

```
Fwd_NumAnchorRefs         1
Bwd_NumAnchorRefs         1
Fwd_NumNonAnchorRefs      0
Bwd_NumNonAnchorRefs      0
Fwd_AnchorRefs            0 6
Bwd_AnchorRefs            0 6
```

The most important parameters that need to be specified in the main configuration file are the name for the bitstream *OutputFile*, the frame rate *FrameRate*, the number of frames to be encoded *FramesToBeEncoded*, the GOP size *GOPSize*. In Example 16, we additionally specified *SearchMode* and *SearchRange* to speed-up the encoder execution.

In multiview coding mode, the generated bitstream can provide temporal scalability using hierarchical B-pictures. The number of supported temporal scalability levels is dependent on the specified GOP size. A so-called group of pictures (GOP) consists of an anchor picture and hierarchically predicted B pictures that are located between the anchor picture of the current GOP and the anchor picture of the previous GOP. In Figure 1, the hierarchical prediction structure is illustrated for a group of 8 pictures.

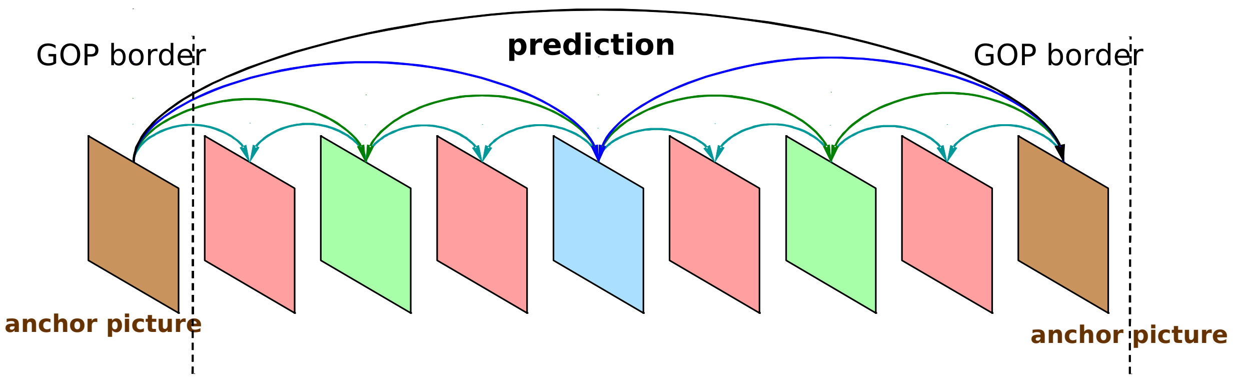

Figure 1: Hierarchical prediction structure for groups of 8 pictures.

The anchor pictures are either intra-coded (e.g. in order to enable random access) or inter-coded by using anchor pictures from another view as reference for motion-compensated prediction. In general it represents the minimal temporal resolution that can be decoded. Furthermore, the anchor pictures can be considered as re-synchronisation points between encoder and decoder. The first picture of a video sequence is always intra-coded as IDR picture and represents a special GOP, which consists of exactly one picture.

The remaining pictures of a GOP are hierarchically predicted. For the example in Figure 1, the picture in the middle (blue) is predicted by using the surrounding key pictures as references. It depends only on the key pictures, and represents the next higher temporal resolution together with the key pictures. The pictures of the next temporal level (green) are predicted by using only the pictures of the lower temporal resolution as references, etc. It is obvious that this hierarchical prediction structure inherently provides temporal scalability.

Given the configuration files in Example 16, the encoder can be started using the call in Example 17. With option "-vf" the configuration file is specified.

Example 17: Example encoder call for multiview coding

```
>H264AVCEncoderLibTestStatic -vf encoderMVC.cfg 0
>H264AVCEncoderLibTestStatic -vf encoderMVC.cfg 2
>H264AVCEncoderLibTestStatic -vf encoderMVC.cfg 1
...and so on in view coding order
```

The encoder should be called separately for each view. The number after the configuration file represents the view_id of the view to be coded. The views should be called in the view coding order to ensure that the reference views are available (reconstructed) before a view begins encoding.

After the encoding of all the views has been completed, a separate bitstream for each view will be created. For the example in Example 16, for the ballroom sequence the bitstreams would be stream_0.264, stream_1.264 etc for all the views. These bitstream then need to be assembled in to a single bitstream before running the decoder. The assembler is explained in detailed in section 2.4.

4 Information for Software Integration

At the JVT meetings, the integration of proposals into the JMVC software is decided. The software coordinators collect information on the integration complexity of all proposals and arrange a schedule for the integration period after the meeting. As input for this procedure, proponents should prepare to give an estimate of the time required for the integration of their proposal by the end of the meeting. The integration time includes the integration work itself and validation testing by the proponent.

The following software integration process is proposed:

- Please check out the latest software version from the CVS.
- It is recommended to run (at least) the short-term validation scripts before integration to make sure that the software passed all tests before you started integration. See section 4.3 for the details how to run the validation scripts.
- Do your integration.
    - Please try hard to stick to the given time slot!
    - Please obey the guidelines in section 4.1.
- In case you might be running into trouble meeting the deadline please contact the software coordinators as soon as possible.
- Re-run the validation tests. To be considered as accepted an implemented tool should "pass" all validation tests. In case the tests fail with a *slight* deviation from the target rate and PSNR values defined in the scripts, please contact the software coordinators to discuss if the target points should be re-adjusted.
- Propose new tests that are designed to ensure that the tool you have implemented is not broken. Therefore, the tests do not need to demonstrate the performance of the tool but should rather validate its operability.
- Update the file "*JMVC_changes.txt*" as well as the software manual.

To verify the validity of the claimed gains of the proposal, proponents are encouraged to provide verification results with the integrated JMVC software for their adopted tool at the next meeting.

4.1 Software integration guidelines and rules

When integrating adopted proposals into the JMVC software, please obey the following basic principles and recommendations:

- **The integrated software shall compile without warnings when using the provided VC6 and VS .NET workspaces as well as linux makefiles (i.e. using an up-to-date g++ compiler).**
- Do not use variable declarations inside the header of for-loops (the scope for for-loops is not correctly supported with all compilers).

- Follow the coding style of the JMVC software. Use 2 (two) spaces for indentation, no tabs.
- Re-use code and integrate functionality as possible. Try to avoid redundant code.
- Do not change the meaning of existing input parameters but define new ones if necessary (and applicable).
- Make sure that new parameters have meaningful default values. Tools should not be switched on by default (if not decided different by the JVT).
- Do not re-structure the output of the compiled binaries (if not decided different by the JVT).
- Please change the JMVC version number (i.e. "_JMVC_VERSION_") macro located in the file "CommonDefs.h" to be inline with your integration tag (e.g. "5.10").

4.2 Information to be provided to the software coordinators group

The following information shall be provided after integration of a proposal into the JMVC software.

- The corresponding proposal numbers (for reference),
- A zipped software package containing only the modified files, but in the correct directory structure *without CVS directories*,
- The output for the current validation scripts when running with the modified software,
- Proposals for modified and/or additional test configurations (also regarding updates of PSNR and rate values) for existing tools.
- Proposals for new short term tests (one or more) that validate the functionality of the new tool.
- A brief list of changes (to maintain the changes file "*JMVC_changes.txt*" in the root directory of the JMVC software – the changes should be directly written to the file "*JMVC_changes.txt*"),
- An updated version of the software manual (when parameters or tools have been added).

4.3 Validation scripts (Not updated)

The validation scripts are written in Perl. In section 4.3.1, a brief overview of the scripts package structure is given. Advices and pre-requisites are formulated in section 4.3.2. Finally, section 4.3.3 described how running the validation scripts.

4.3.1 Structure

The root directory of the validation scripts contains the following file and sub-directories:

- *run.pm (file):* which is the main Perl script to be run in order to validate/evaluate JMVC performances
- *Tools (dir):* which contains additional Perl scripts.
- *SimuDataBase (dir):* which contains the Simulations (i.e. tests sets) data base to be run for the validation process
- *PacketLossSimulator (dir):* which contains the source code of a dummy packet loss simulator to be used for validation tests related to Error Concealment.

Three tests sets have been designed:

- *Short_term:* represents a set of several short tests (on a few number of images) which aim at validating most of the tools integrated in the JMVC software. Crossing tests running various tools together are also defined. Each time a tool is integrated, the proponents are invited to propose new dedicated short tests. The running duration of this test set would be less than 3 hours.
- *Long_term*: represents a set of long tests that target to asset the sanity of the software over time and face to more complex encoding/decoding scenarios. For the time being, only one simulation is present

- *AVC_Conformance*: represents a set of AVC conformance tests which aim at validating the conformance of the JMVC decoder. The numbering of the conformance tests corresponds to the classification of the conformance bitstreams in the ITU-T Recommendation H.264.1 "Conformance specification for H.264 advanced video coding".

From the AVC conformance test set, only the tests 6.6.1, 6.6.11, and Hierarchical need to be run. All other AVC conformance tests are expected to fail, since not all AVC features are implemented in the current JMVC software version.

4.3.2 Before running the scripts

Before running the *Short_term* and *Long_term* tests sets, the user will have to download YUV sequences available at ftp.tnt.uni-hannover.de/pub/svc/testsequences/. The required sequences the following: BUS_352x288_30.yuv, MOBILE_352x288_30.yuv, FOOTBALL_352x288_30.yuv (*Short_term*) and CREW_704x576_30.yuv (*Long_term*).

To run the validation tests related to the Error Concealment features. You need first to generate the *PacketLossSimulatorStatic* executable. The PacketLossSimulator project is located at the root of the *Validation* directory.

Before running the *AVC_Conformance* test set, the user will have to download the conformance bitstreams and YUV sequences available at http://ftp3.itu.ch/av-arch/jvt-site/draft_conformance/. Then, you should use the *dump.pm* Perl script (located in *SimuDataBase/AVC_Conformance* directory) in order to copy (or remove) the sequences and bitstreams to their right places. Example 18 illustrates the *dump.pm* usage.

Example 18: Using the dump.pm script

```
USAGE: dump.pm
--------------
[-c]     : to copy the "conformance sequences and bitstreams" in
           the corresponding simus directories.
[-r]     : to remove the "conformance sequences and bitstreams" of
           each simus directories.
[-simu <name_simu1>...<name_simuN> ] : name of the simulations to copy/remove.
[-data <yuv_streams_directory>]      : name of the directory containing the
                                       "conformance sequences and bitstreams".
[-which] : print the name of "conformance sequences and bitstreams" to be used.
[-u]     : Usage.
```

In order to find out which sequences and bitstreams are necessary in order to run a given *AVC_Conformance* test, the *–which* option can be used. Example 19 illustrates the way to use the *dump.pm* script in order to know which conformance bitstreams and sequences are needed when running test named *6.6.1*.

Example 19: Knowing the necessary conformance bitstreams and sequences

```
> perl dump.pm -simu 6.6.1 -which
```

Assuming the user has downloaded the conformance sequences and bitstreams in a directory named *ConformanceData*. Assuming that the user intends to copy the data related to test sets named *6.6.1* and *6.6.11* to the right directories. Example 20 illustrates the way the *dump.pm* script should be run.

Example 20: Copying conformance data using the dump.pm script

```
> perl dump.pm -c -data ./ConformanceData –simu 6.6.1 6.6.11
```

The binaries directory must contain the following executables:,*H264AVCEncoderLibTestStatic*, *H264AVCDecoderLibTestStatic, BitStreamExtractorStatic*, *QualityLevelAssignerStatic*, *PSNRStatic*, *DownConvertStatic* and, *PacketLossSimulatorStatic.*

4.3.3 Running validation scripts

To run the so-called "validation scripts" the user shall launch the main script *run.pm* with the appropriate arguments (see Example 21). By default, the *Short_term* test set will be run assuming *./bin* as binaries directory and *./orig* as YUV sequences directory.

Example 21: Using the run.pm script

```
USAGE: run.pm
--------------
[-SimusetName  [<SimuName1>...<SimuNameN>]]
    : to specify that thesimulations-set called
      "SimusetName" must be run (default value: Short_term)
    : "SimusetName" must be the name of a sub-directory of
      the SimuDataBase directory

  NOTE 0: You can specify the name of the simulations
          you want to run inside a given
          simulations-set by specifying <SimuName1>...<SimuNameN>.
  NOTE 1: You can just specify a prefix of the SimusetName set.

[-bin <bin_directory>]   :to specify the
                          bin directory location (default value: ./bin)
[-seq <orig_directory>]  :to specify the sequences
                          directory location     (default value: ./orig)
[-v]  : Version number
[-u] : Usage
```

The name of the simulations set to be validated can be also specified by using *–SimusetName* where *SimusetName* represents the name of the targeted test set. So far, the possible tests set name are *Short_term*, *Long_term* and *AVC_Conformance*. To simplify the use of such an option the user may only specify a prefix of the test set name to be validated. Example 22 illustrates the use of the run.pm script when the user intends to validate the *Long_term* tests set.

Additionnally to the -*SimusetName* the user may specify explicitly the name of the tests to run in the considered tests set. To specify it ,the full names of the simulations (tests) to be run separated by white spaces should be given as follows *SimuName1 SimuName2 ...SimuNameN.*

The binaries directory as well as the input YUV sequences directory could be specified by using respectively the options *–bin* and *–seq*.

Example 22: Validating the Long_term tests set

```
> perl run.pm -Long_term

is equivalent to

> perl run.pm -L
```

Example 23 illustrates the command line when user intends to validate the tests named *CAVLC*, *ESS* and, *T1* of the simulations set *Short_term* assuming *../../bin* as binaries directory and *../../orig* as YUV sequences directory.

Example 23: Validating the tests CAVLC, ESS and T1 of the Short_term set

```
> perl run.pm -S CAVLC ESS T1 -seq ../../orig -bin ../../bin
```

A working directory named *SimuRun* is created (if non existing), during the validation scripts running. For each run tests a detailed log file as well as a global log file would be created at the root of the *SimuRun* directory. To be considered as validated, the global log file of a given simulation shall contain only "*passed*" messages. If "*failed*" or "*no results*" messages occur, it can result from a bad validation scripts usage or worse, a broken JMVC tool. Example 24 illustrates the log file *CAVLC_Global.log* assuming a successful validation of the *CAVLC* test.

Example 24: CAVLC_Global.log file assuming a successful validation the test CAVLC.

```
==================================================
 Run simu CAVLC:
-------------------
 Load Simu.............. ok
 Create Sequences....... ok
 Encode................. ok
 Run Tests..............

-----------------------------------------------
L0 :: (176x144, 15) -> 268.2 - 34.47
-----------------------------------------------
      Rate   (265.4400)          Passed
      PSNR   (34.4708)           Passed
      Encoder/Decoder match            Passed
-----------------------------------------------
L1 :: (352x288, 30) -> 1565 - 33.88
-----------------------------------------------
      Rate   (1563.3680)         Passed
      PSNR   (33.9121)           Passed
      Encoder/Decoder match            Passed
-----------------------------------------------
Temp Base :: (352x288, 3.75) -> 805 - 37.05
-----------------------------------------------
      Rate   (801.6225)          Passed
      PSNR   (37.0608)           Passed
ok
```