JSVM Software Manual

Version: JSVM 6.8.2 (CVS tag: JSVM_6_8_2)

Last update: October 13, 2006

Summary:

This document contains a detailed description of the usage and configuration of the JSVM (Joint Scalable Video Model) software for the Scalable Video Coding (SVC) 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 spatial, SNR and combined scalability scenarios.

Guidelines for the integration and validation of new tools in the software are provided.
4 Information for Software Integration .................................................................................................................. 62
4.1 Software integration guidelines and rules ........................................................................................................ 63
4.2 Information to be provided to the software coordinators group ................................................................. 63
4.3 Validation scripts.............................................................................................................................................. 64
   4.3.1 Structure .................................................................................................................................................. 64
   4.3.2 Before running the scripts ....................................................................................................................... 64
   4.3.3 Running validation scripts ....................................................................................................................... 65
1 General Information

The JSVM (Joint Scalable Video Model) software is the reference software for the Scalable Video Coding (SVC) 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 SVC project is still under development, the JSVM Software as is also under development and changes frequently.

The JSVM software is written in C++ and is provided as source code. Section 1.1 describes how the JSVM 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 JSVM software can be build on Win32 and Linux platforms, and section 1.4 gives basic information about the binaries that are contained in the JSVM software package.

1.1 Accessing the latest JSVM Software

In order to keep track of the changes in software development and to always provide an up-to-date version of the JSVM software, a CVS server for the JSVM 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 JSVM software server is restricted to the JSVM software coordinators group.

<table>
<thead>
<tr>
<th>Table 1: CVS access parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication: pserver</td>
</tr>
<tr>
<td>host address: garcon.ient.rwth-aachen.de</td>
</tr>
<tr>
<td>path: /cvs/jvt</td>
</tr>
<tr>
<td>user name: jvtuser</td>
</tr>
<tr>
<td>password: jvt.Amd.2</td>
</tr>
<tr>
<td>module name: jsvm</td>
</tr>
</tbody>
</table>

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

Example 1: Accessing the JSVM 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 jsvm

In Example 2, it is shown how a specific JSVM software version – specified by a tag (JSVM_4_5 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 JSVM software version with the tag JSVM_4_5 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 JSVM_4_5 jsvm

1.2 Structure of the CVS Repository

After accessing the JSVM software as described in section 1.1, a folder jsvm 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 JSVM contains all files that are required for building and running the software.

The folder JSVM/H264Extension/src is structured into sub-folders for libraries, test projects, and tools. It contains all source and include files for the JSVM 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 `JSVM/H264Extension/include`.

A log file describing the (main) changes from one JSVM software version to the next is given by `changes.txt`. Note that this log file starts with the JSVM version 4.0 (CVS tag JSVM_4_0). Earlier changes haven’t been tracked.

### Table 2: Structure of the CVS repository for the JSVM software

<table>
<thead>
<tr>
<th>folder</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>location of binaries after building the software</td>
</tr>
<tr>
<td></td>
<td>More information about the binaries are given in section 1.4.</td>
</tr>
<tr>
<td>lib</td>
<td>location of libraries after building the software</td>
</tr>
<tr>
<td></td>
<td>More information regarding the libraries are given in section 1.4.</td>
</tr>
<tr>
<td>JSVM</td>
<td>source code and project files for the JSVM software</td>
</tr>
<tr>
<td></td>
<td>All files that are required for building and using the JSVM software</td>
</tr>
<tr>
<td></td>
<td>are contained in this folder.</td>
</tr>
<tr>
<td>JSVM/H264Extension/build</td>
<td>workspaces and makesfiles</td>
</tr>
<tr>
<td></td>
<td>Workspaces are provided for Microsoft Visual Studio 6 and</td>
</tr>
<tr>
<td></td>
<td>Microsoft Visual Studio .NET. Makefiles are provided for Linux.</td>
</tr>
<tr>
<td>JSVM/H264Extension/data</td>
<td>basic examples of encoder configuration files</td>
</tr>
<tr>
<td></td>
<td>These examples are not guaranteed to be complete or to contain all</td>
</tr>
<tr>
<td></td>
<td>available configuration parameters. The examples only serve as a</td>
</tr>
<tr>
<td></td>
<td>basis for writing encoder configuration files.</td>
</tr>
<tr>
<td>JSVM/H264Extension/include</td>
<td>includes files that are required by other libraries inside the JSVM</td>
</tr>
<tr>
<td></td>
<td>project</td>
</tr>
<tr>
<td></td>
<td>Only include files for classes that are required by other libraries or</td>
</tr>
<tr>
<td></td>
<td>test projects should be placed into this folder. This folder contains</td>
</tr>
<tr>
<td></td>
<td>a subfolder for the frequently used library H264AVCCommonLib. Include</td>
</tr>
<tr>
<td></td>
<td>files of classes that are only required inside a library or test project</td>
</tr>
<tr>
<td></td>
<td>are contained in the library or test project folder in</td>
</tr>
<tr>
<td></td>
<td>JSVM/H264Extension/src.</td>
</tr>
<tr>
<td>JSVM/H264Extension/src</td>
<td>include and source files for all libraries and test projects</td>
</tr>
<tr>
<td></td>
<td>This folder contains all source files and the include files that are</td>
</tr>
<tr>
<td></td>
<td>only required inside a library or test project. This folder is organized</td>
</tr>
<tr>
<td></td>
<td>in appropriate sub-folders.</td>
</tr>
<tr>
<td>JSVM0-config-samples</td>
<td>usage examples</td>
</tr>
<tr>
<td></td>
<td>This folder contains usage examples with configuration files, scripts</td>
</tr>
<tr>
<td></td>
<td>as well as some text descriptions. These usage examples are not</td>
</tr>
<tr>
<td></td>
<td>updated with each change of the JSVM software, and hence they are not</td>
</tr>
<tr>
<td></td>
<td>guaranteed to be up-to-date.</td>
</tr>
<tr>
<td>MVC-Configs</td>
<td>configuration examples for Multiple View Coding</td>
</tr>
<tr>
<td></td>
<td>This folder contains examples including configuration files and</td>
</tr>
<tr>
<td></td>
<td>scripts for using the JSVM software for Multi View Coding (MVC).</td>
</tr>
<tr>
<td>Validation</td>
<td>validation scripts for checking the JSVM software</td>
</tr>
<tr>
<td></td>
<td>Perl scripts for validating the JSVM software are provided in this</td>
</tr>
<tr>
<td></td>
<td>folder. These scripts shall be run after each modification of the</td>
</tr>
<tr>
<td></td>
<td>JSVM software. They have been designed to check the functionality</td>
</tr>
<tr>
<td></td>
<td>of integrated tools. More information on how to run these scripts are</td>
</tr>
<tr>
<td></td>
<td>given in 4.3.</td>
</tr>
<tr>
<td>changes.txt</td>
<td>changes log file</td>
</tr>
<tr>
<td></td>
<td>This file described the (main) changes from one CVS version to the</td>
</tr>
<tr>
<td></td>
<td>next. It starts with JSVM version 4.0 (CVS tag: JSVM_4_0).</td>
</tr>
</tbody>
</table>

### 1.3 Building the JSVM software

It shall be possible to build the JSVM 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 JSVM 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 `JSVM/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 `JSVM/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/BitStreamExtractorStatic.exe
bin/BitStreamExtractorStaticd.exe
bin/DownConvertStatic.exe
bin/DownConvertStaticd.exe
bin/FixedQPEncoderStatic.exe
bin/FixedQPEncoderStaticd.exe
bin/H264AVCDecoderLibTestStatic.exe
bin/H264AVCDecoderLibTestStaticd.exe
bin/H264AVCEncoderLibTestStatic.exe
bin/H264AVCEncoderLibTestStaticd.exe
bin/MCTFPreProcessorStatic.exe
bin/MCTFPreProcessorStaticd.exe
bin/PSNRStatic.exe
bin/PSNRStaticd.exe
bin/QualityLevelAssignerStatic.exe
bin/QualityLevelAssignerStaticd.exe
bin/SIPAnalyser.exe
bin/SIPAnalyserd.exe

===== libraries =====
lib/H264AVCCCommonLibStatic.lib
lib/H264AVCCCommonLibStaticd.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 JSVM/H264Extension/build/linux and the corresponding sub-folders. Supposing that the current folder is the main folder jsvm of the JSVM repository (see section 1.2), the commands specified in Example 4 should be executed to build all project files.

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

```
cd JSVM/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

```text
===== binaries =====
bin/BitStreamExtractorStatic
bin/BitStreamExtractorStaticd
bin/DownConvertStatic
bin/DownConvertStaticd
bin/FixedQPEncoderStatic
bin/FixedQPEncoderStaticd
bin/H264AVCDecoderLibTestStatic
bin/H264AVCDecoderLibTestStaticd
bin/H264AVCEncoderLibTestStatic
bin/H264AVCEncoderLibTestStaticd
bin/MCTFPreProcessorStatic
bin/MCTFPreProcessorStaticd
bin/PSNRStatic
bin/PSNRStaticd
bin/QualityLevelAssignerStatic
bin/QualityLevelAssignerStaticd
bin/SIPAnalyser
bin/SIPAnalyserd

===== 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
```

1.4 Information on binaries and libraries

Table 3 and Table 4 give information on the libraries and executables that are contained in the JSVM 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 0 several examples for using the JSVM software are given in the form of a tutorial.

### Table 3: Libraries provided by the JSVM software

<table>
<thead>
<tr>
<th>library</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H264AVCCCommonLibStatic</td>
<td>common lib</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>H264AVCEncoderLibStatic</td>
<td>encoder lib</td>
</tr>
<tr>
<td></td>
<td>This library provides classes that are only used by the encoder. For example, it includes classes for motion estimation, mode decision, and entropy encoding.</td>
</tr>
<tr>
<td>H264AVCDecoderLibStatic</td>
<td>decoder lib</td>
</tr>
<tr>
<td></td>
<td>This library provides classes that are only used by the decoder. For example, it includes classes for entropy decoding and bit-stream parsing.</td>
</tr>
<tr>
<td>H264AVCVideoIoLibStatic</td>
<td>io lib</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

### Table 4: Executables provided by the JSVM software

<table>
<thead>
<tr>
<th>executable</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DownConvertStatic</td>
<td>resampler</td>
</tr>
<tr>
<td></td>
<td>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 2.1.</td>
</tr>
<tr>
<td>H264AVCEncoderLibTestStatic</td>
<td>AVC/SVC encoder</td>
</tr>
<tr>
<td></td>
<td>The encoder can be used for generating single-layer (AVC) or scalable (SVC) bit-streams. More information on using the encoder are provided in section 2.2.</td>
</tr>
<tr>
<td>H264AVCDecoderLibTestStatic</td>
<td>SVC decoder</td>
</tr>
<tr>
<td></td>
<td>The decoder can be used for decoding AVC or SVC bit-streams and reconstructing raw video sequences. More information on using the decoder are provided in section 2.3.</td>
</tr>
<tr>
<td>BitStreamExtractorStatic</td>
<td>bit-stream extractor</td>
</tr>
<tr>
<td></td>
<td>The bit-stream extractor can be used for extracting sub-bitstreams with a lower spatio-temporal resolution and/or bit-rate from a global scalable (SVC) bit-stream. More information on using the bit-stream extractor are provided in section 2.4.</td>
</tr>
<tr>
<td>QualityLevelAssignerStatic</td>
<td>quality level assigner</td>
</tr>
<tr>
<td></td>
<td>The quality level assigner can be used for generating a bit-stream with additional quality layer information given a scalable bit-stream. Beside the additional quality layer information input and output bit-stream are identical. More information on using the quality layer assigner are provided in section 2.5.</td>
</tr>
<tr>
<td>MCTFPreProcessor</td>
<td>MCTF pre-processor</td>
</tr>
<tr>
<td></td>
<td>The MCTF pre-processing tool can be used for pre-filtering image sequences. More information on using the MCTF pre-processor are provided in section 2.6.</td>
</tr>
<tr>
<td>PSNRStatic</td>
<td>PSNR tool</td>
</tr>
<tr>
<td></td>
<td>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 bit-stream. More information on using the PSNR tool are provided in section 2.6.</td>
</tr>
</tbody>
</table>
2 Usage and configuration of the JSVM software

This section provides information on usage and configuration of the binaries contained in the JSVM software package. Additionally, examples for using the software are provided in section 0.

2.1 Resampler “DownConvertStatic”

The resampler can be operated in different modes. Various filters and tools are supported. An overview of the supported features and parameters is given in Example 6. The common parameters are described in section 2.1.1. Then, the different resampling modes and methods are described in section 2.1.2. Section 2.1.3 describes parameters that are specific to Extended Spatial Scalability (ESS), the cropping window and chroma phase shift features.

Example 6: Using the resampler

```
DownConvertStatic <win> <hin> <in> <wout> <hout> <out>
    [method] [t] [skip] [frms] ]]
    [[-crop <args>] [-phase <args>]]

win     : input width  (luma samples)
hin     : input height (luma samples)
in      : input file
wout    : output width  (luma samples)
hout    : output height (luma samples)
out     : output file
--------------------------- OPTIONAL ---------------------------
method  : rescaling methods (default: 0)
0: normative upsampling
   non-normative downsampling (JVT-R006)
1: dyadic upsampling (AVC 6-tap (1/2 pel) on odd luma samples
   dyadic downsampling (MPEG-4 downsampling filter)
2: crop only
3: upsampling (Three-lobed Lanczos-windowed sinc)
4: upsampling (JVT-0041: AVC 6-tap (1/2 pel) + bilinear 1/4 pel

t       : number of temporal downsampling stages (default: 0)
skip    : number of frames to skip at start (default: 0)
frms    : number of frames wanted in output file (default: max)

-------------------------- OVERLOADED --------------------------
-crop <type> <parameters>
type     : 0: Sequence level,  1: Picture level
params   : IF Sequence level: <x_orig> <y_orig> <crop_width> <crop_height>
          cropping window origin (x,y) and dimensions (width and height)
          IF Picture level: <crop_file>
          input file containing cropping window parameters.
          each line has four integer numbers separated by a comma
          as following: "x_orig, y_orig, crop_width, crop_height"
          for each picture to be resampled;
-phase <in_uv_ph_x> <in_uv_ph_y> <out_uv_ph_x> <out_uv_ph_y>
in_uv_ph_x : Input chroma phase shift in horizontal direction (default:-1)
in_uv_ph_y : input chroma phase shift in vertical direction (default: 0)
out_uv_ph_x: output chroma phase shift in horizontal direction (default:-1)
out_uv_ph_y: output chroma phase shift in vertical direction (default: 0)
```
2.1.1 Common parameters

The parameters \( \text{win} \) and \( \text{hin} \) specify the width and height of the input images in luma samples. The parameters \( \text{wout} \) and \( \text{hout} \) represents the width and height of the output images in luma samples. The parameters \( \text{in} \) and \( \text{out} \) specify the filenames of the input and output video sequences. Then according to the specified dimensions, the tool will be used either in \textit{upsampling} or \textit{downsampling} mode. By default, the upsampling method used is the SVC normative one (see section 2.1.2) and the downsampling method is based on the filters set proposed in JVT-R006 (see section 2.1.2). Different resampling methods can be specified by setting the \textit{method} argument as described in the following section.

The parameters \( t \), \( \text{skip} \), and \( \text{frms} \) are optional. The parameter \( t \) specifies the number of temporal downsampling stages. Only every \( (1 << t) \) image is written to the output file. It is not possible to specify negative values (i.e. temporal upsampling) for the parameter \( t \). With the parameter \( \text{skip} \) it can be specified how many frames at the beginning of the input sequence shall be skipped. And with the optional parameter \( \text{frms} \), a maximum number of output frames that shall be produced can be specified.

2.1.2 Resampling methods

The parameter \textit{method} specifies the method/filters to be used for the resampling process. Four values (0, 1, 3 and 4) are supported for the upsampling and only two for the downsampling (0 and 1). The specific value of 2 is reserved to the \textit{crop only} process.

2.1.2.1 Upsampling

When \textit{method} is set to 0, the SVC normative upsampling method designed to support the Extended Spatial Scalability is applied. It is based on a set of 6-taps filters. These integer-based 6-tap filters are derived from the Lanczos-3 filter. Any inter-layer scaling ratios, which can also be different in horizontal and vertical, are supported.

When \textit{method} is set to 1, only dyadic rescaling ratios are supported. The upsampling is realized via several dyadic stages. By default, in each stage, every second sample in horizontal and vertical direction is presented by the samples of the input image, and the missing luma samples are interpolated using the AVC half-sample interpolation filter \( \{1, -5, 20, 20, -5, 1\} / 32 \). The missing chroma samples are interpolated using the “very simple” filter \( \{16, 16\}/32 \).

Note that the ratio between the input and output dimensions shall be a power of 2.

When \textit{method} is set to 3, the upsampling method is achieved applying the three-lobed Lanczos-windowed sinc functions. Any inter-layer scaling ratios, which can also be different in horizontal and vertical, are supported.

When \textit{method} is set to 4, a combination of the AVC half-sample filter (see above) and a bi-linear filter is used.

2.1.2.2 Downsampling

When \textit{method} is set to 0, the downsampling method based on the Sine-windowed Sinc-function is applied. A set of seven filters has been designed to support the extended range of spatial scaling ratio required by ESS. Any inter-layer scaling ratios, which can also be different in horizontal and vertical, are supported.

When \textit{method} is set to 1, we switch on dyadic downsampling mode. As for the dyadic upsampling the downsampling is done via several stages, where in each stage a scaling factor of 0.5 is applied in both horizontal and vertical direction. In each stage the input images are filtered with the MPEG-4 downsampling filter \( \{2, 0, -4, -3, 5, 19, 26, 19, 5, -3, -4, 0, 2\} / 64 \) and the output is presented by every
second sample (in horizontal and vertical direction) of the filtered images. Note that the ratio between the input and output dimensions shall be a power of 2.

In Example 7, the command line call for down-sampling a 4CIF (704x576 samples) sequence with a frame rate of 60Hz to a QCIF (176x144 samples) sequence with a frame rate of 15Hz is illustrated. Please note that this example uses method 1 which is not the recommended method for the simulations with the JSVM (see section 3.1). In Example 8, the resampling of a CIF (352x288 samples) 30Hz sequence to a 528x432 15Hz sequence using the SVC normative up-sampler is illustrated.

Example 7: Down-sampling a 4CIF 60Hz sequence to a QCIF 15Hz sequence using the dyadic method.

```
DownConvertStatic 704 576 4CIF60.yuv 176 144 QCIF15.yuv 1 2
```

Example 8: Resampling of a CIF 30Hz sequence to a 528x432 15Hz sequence using the normative up-sampler.

```
DownConvertStatic 352 288 CIF30.yuv 528 432 528x432_15.yuv 0 1
```

### 2.1.3 Specific features

Two specific resampling features originally introduced for the ESS are supported: cropping window and chroma phase shift management.

#### 2.1.3.1 Cropping window

The use of the cropping features allows generating the output sequence as a cropped area (window) of the input sequence.

Cropping parameters are specified by using the `–crop` option. Cropping parameters can be specified either for the entire sequence or for each picture of the sequence. Thus, after the `–crop` option the crop type set to 0 for sequence level or to 1 for picture level shall be specified.

If sequence level is chosen, then four parameters `x_orig y_orig crop_width crop_height` will follow the type. Where `x_orig` and `y_orig` represent the coordinates of the upper left corner of the cropping window and `crop_width` and `crop_height` represent respectively its width and height.

If picture level is chosen, the crop type argument is followed by argument `crop_file` which specifies the filename of the file containing the cropping parameters. Each line of the file is made up of four integers separated by commas as following: “`x_orig, y_orig, crop_width, crop_height`”. The meaning of these parameters is the same as above. In case, the file contains less lines than the number of pictures in the output sequence, the cropping parameters of the last line will be used for the following pictures. Example 9 illustrates the structure of a cropping parameter file.

**Example 9: Structure of the cropping parameter file**

```
<x_orig> <y_orig> <crop_width> <crop_height>  // parameters for frame 0
<x_orig> <y_orig> <crop_width> <crop_height>  // parameters for frame 1
<x_orig> <y_orig> <crop_width> <crop_height>  // parameters for frame 2
<x_orig> <y_orig> <crop_width> <crop_height>  // parameters for frame 3
...
```

Example 10 illustrates the resampling of a 720p (HD) sequence with a resolution of 1280x720 samples and a frame rate of 50Hz to an SD sequence with a resolution of 720x576 samples and a frame rate of 25Hz. Note that with this example, the aspect ratio of the samples is kept, but the left and right border of the HD sequence in 16:9 format are cut off in order to obtain an SD sequence in 5:4 format. First an area of 900x720 samples if cropped out of the middle of an HD pictures (horizontal offset of 190 samples), and then these cropped pictures are downsampled with a resampling factor of 4/5 in both horizontal and vertical direction.
Example 10: Resampling of a HD sequence (1280x720,50Hz) to an SD sequence (720x576,25Hz)

```
DownConvertStatic.exe 1280 720 720p50.yuv 720 576 SD25.yuv 0 –crop 0 crop.txt 1
```

content of crop.txt:
190 0 900 720

Note that the resampler can be used in crop only mode by setting the method argument to 2. In case no –crop option is specified, the width and height of the cropping window are the output parameters wout and hout.

2.1.3.2 Chroma phase shift management

By using the -phase option followed by the four parameters in_uv_ph_x, in_uv_ph_y, out_uv_ph_x and out_uv_ph_y, the horizontal and vertical phase shift the chroma components in comparison to the luma component can be specified in quarter luma samples for the input sequence as well the horizontal and vertical chroma phase shift of the output sequence can be specified. The allowed value for this parameter should be in the range of -1 to 1, inclusive.

2.2 Encoder “H264AVCEncoderLibTestStatic”

The encoder can be used for generating AVC or SVC bit-streams. The basic encoder call is illustrated in Example 11. At this mcfg represents the filename of the main configuration file. The main configuration file shall be specified for each encoder call. In addition to the configuration file, several command line options can be specified.

```
Example 11: Using the encoder
H264AVCEncoderLibTestStatic.exe -pf <mcfg> [command line options]
```

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. Special care has to be taken when running the encoder in scalable mode, since the coding efficiency generally decrease with the supported scalability options. 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 0.

The encoder can be run in two different coding modes: single-layer coding mode and scalable coding mode. Although single-layer bit-stream can also be generated in the scalable coding mode, the single-layer coding mode provides more flexibility but lacks the support generating scalable bit-streams. The encoding mode is specified by the parameter MVCMode inside the main configuration file. When this parameter is not present or equal to 0, the encoder is run in single-layer coding mode; otherwise, the encoder is operated in scalable coding mode. The configuration file parameters and command line option for the single-layer coding mode are described in section 2.2.1, while the configuration file parameters and command line options for the scalable coding mode are described in section 2.2.2.

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.

2.2.1 Single-layer coding mode

The encoder is run in single-layer coding mode, when the configuration file contains the parameter MVCMode and this parameter is set equal to 1. The configuration file parameters for the single-layer coding mode are described in section 2.2.1.1, and the command line options are described in section 2.2.1.3. When the encoder is run in single-layer mode, an AVC compatible bit-stream is generated.
2.2.1.1 Configuration file

All available configuration file parameters for the single-layer mode together with a brief description are summarized in Example 12. Additional information about the configuration parameters, including default values, is given below.

Example 12: Encoder configuration file in single-layer coding mode

```
# JSVM Encoder configuration File in MVC mode

#----------------------------- GENERAL ------------------------------------------
MVCMode 1 # must be one for MVC simulations
InputFile input.yuv # input file
OutputFile stream.264 # bitstream file
ReconFile rec.yuv # reconstructed file
SourceWidth 352 # input frame width
SourceHeight 288 # input frame height
FrameRate 25.0 # frame rate [Hz]
FramesToBeEncoded 300 # number of frames

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

#----------------------------- STRUCTURE --------------------------------------
DPBSize 16 # decoded picture buffer in frames
NumRefFrames 16 # maximum number of stored reference frames
Log2MaxFrameNum 11 # specifies max. value for frame_num (4..16)
Log2MaxPocLsb 7 # specifies coding of POC's (4..15)
SequenceFormatString A0L0*74{P3L0B1L1b0L2b2L2}*1{B1L1b0L2b2L2} # coding structure
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)
```

MVCMode

*Flag (0 or 1), default: 0*
Specifies whether the encoder is run in single-layer coding mode, which is also referred to as Multiview coding mode, since this mode was implemented to support multiview coding. When MVCMode is equal to 1, the scalability tools can not be used, but the coding structure is not restricted to dyadic prediction structures (cp. section 2.2.1.2). Note that dependent on the value of MVCMode several configuration parameters have no influence. When MVCMode is equal to 0, the encoder is run in scalable coding mode, which is described in section 2.2.2.

**InputFile**

*String, default: “in.yuv”*

Specifies the filename of the original raw video sequence to be encoded.

**OutputFile**

*String, default: “test.264”*

Specifies the filename of the bit-stream to be generated.

**ReconFile**

*String, default: “rec.yuv”*

Specifies the filename of the coded and reconstructed input sequence. This sequence is provided for debugging purposes.

**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.

**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 bit-stream. 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.

**DPBSize**
Unsigned Int, default: 1
Specifies the minimum size of the decoded picture buffer (DPB) in frames. Depending on this parameter, the Level for encoding a video sequence is selected.

**NumRefFrames**

Unsigned Int, default: 1
Specifies the maximum number of reference frames that are stored in the DPB and can be referenced via Inter prediction of a following frame. The parameter also specifies the value of the syntax element num_ref_frames. NumRefFrames shall not be greater than DPBSize.

**Log2MaxFrameNum**

Unsigned Int, default: 4
Specifies the maximum value of the syntax elements frame_num. It also specifies the value of the syntax element log2_max_frame_num_minus4. Log2MaxFrameNum shall be in the range of 4 to 16, inclusive. Note that this parameter shall be large enough to allow an encoding/decoding of the sequence structure specified by SequenceFormatString. A greater value of Log2MaxFrameNum also increases the robustness against packet losses. When the value of Log2MaxFrameNum is too small in an error prone environment, the number of actual missing frames cannot be detected.

**Log2MaxPocLsb**

Unsigned Int, default: 4
Specifies the number of bits that are used for transmitting the syntax elements pic_order_cnt_lsb. It also specifies the value of the syntax element log2_max_pic_order_cnt_lsb_minus4. Log2MaxPocLsb shall be in the range of 4 to 15, inclusive. Note that this parameter shall be large enough to allow an encoding/decoding of the sequence structure specified by SequenceFormatString. A greater value of Log2MaxPocLsb also increases the robustness against packet losses. When the value of Log2MaxPocLsb is too small in an error prone environment, the output order of decoded pictures cannot be correctly determined.

**SequenceFormatString**

String, default: “A0*n{P0}”
Specifies the coding structure for the video sequence. This parameter allows a very flexible configuration, more information about using this parameter are provided in section 2.2.1.2.

**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 assignment of layers to individual frames is specified by the parameter SequenceFormatString as described in section 2.2.1.2. The quantization parameter of a frame of layer X is calculated by QP = BasisQP + DeltaLayerXQuant.

**MaxRefIdxActiveBLX** (with X being replaced by 0 or 1)

Unsigned Int, default: 1
Specifies the maximum number of active entries in the reference picture list X for B pictures.

**MaxRefIdxActiveP**

Unsigned Int, default: 1
Specifies the maximum number of active entries in the reference picture list for P pictures.

**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 applied. 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.
2.2.1.2 Sequence format string

The sequence coding structure together with Memory Management Control Operation (MMCO) and Reference Picture List Reordering (RPLR) commands is specified in the encoder configuration file via a formatted string `SequenceFormatString`.

2.2.1.2.1 Syntax of the sequence format string

The general syntax of the sequence structure string is given by

```
sequence structure:   GSP

where GSP is a string that specifies a general sequence part. It generally consists of an arbitrary combination of frame sequence parts (FSP) and/or further general sequence parts (GSP) that can be optionally enclosed in curly brackets preceded by an asterisk and a syntax element REP specifying the number of repetitions of the sub-sequence structure inside the curly brackets:

GSP:       [REP{]GSP|FSP[GSP|FSP]…[GSP|FSP][}]

The syntax element REP specifies the number of repetitions of the sub-sequences enclosed in curly brackets, it can be either a decimal number or the character ‘n’, which specifies that the sub-sequence structure encloses in curly brackets in repeated forever:

REP:       [(decimal number)|n]

A frame sequence part (FSP) consists of a combination of frame descriptions (FDES) that can be optionally enclosed in curly brackets preceded by an asterisk and the syntax element REP specifying the number of repetitions of the sub-sequence structure enclosed inside the curly brackets in the same manner as for the general sequence part (GPS):

FSP:        [REP{]FDES[FDES]…[FDES][]}

A frame description (FDES) specifies all parameters needed by the encoder for encoding a given frame of a video sequence and updating the decoded picture buffer (DPB). A frame specification (FDES) consists of a frame coding type (TYPE) specifying the slice coding type and whether the frame is marked as “used for reference”, a frame index offset (OFFSET) specifying the difference of the frame indices of the frame to encode and the first frame (in input order) of the current frame sequence part (FSP) inside the input video sequence, and optionally up to two lists of reference picture reordering commands and/or a list of memory management control operation commands:

FSPEC:      (TYPE)(OFFSET)[Layer][RPLR-List0[RPLR-List1]][MMCO-List]

The frame coding type (TYPE) specifies the slice coding type for the frame and it further determines whether the frame is marked as “used for reference” after storing in the decoded picture buffer. The following values are possible:
TYPE: A|I|P|B|i|p|b

A: The first picture of the frame is coded as IDR picture (the second picture is coded as P picture and marked as “used for reference”).
I: The picture(s) is/are coded as I picture(s) and marked as “used for reference”.
P: The picture(s) is/are coded as P picture(s) and marked as “used for reference”.
B: The picture(s) is/are coded as B picture(s) and marked as “used for reference”.
i: The picture(s) is/are coded as I picture(s) and marked as “unused for reference”.
p: The picture(s) is/are coded as P picture(s) and marked as “unused for reference”.
b: The picture(s) is/are coded as B picture(s) and marked as “unused for reference”.

The frame index offset (OFFSET) is a decimal number, which specifies the frame index of the given frame inside the given input video sequence:

OFFSET: (decimal number)

This frame index is not given as absolute number, but as difference between the frame index of the current frame and the frame index of the first frame (in input order) of the current frame sequence part (FSP). If a frame sequence part consists of N frames, each frame index offset (OFFSET) must greater or equal to 0 and less than N, and all frame index offsets inside the frame sequence part must be different.

The layer specification starts with the character ‘L’ followed by a decimal number.

Layer: L(LayerNumber)

The integer LayerNumber specifies the assignment of a picture to a specific layer. LayerNumber shall be in the range of 0 to 5, inclusive. Pictures are grouped into layers, for assigning them a specific quantization parameter is described in section 2.2.1.1. When the sub-string Layer is not present, the LayerNumber is inferred to be equal to 0.

The lists of reference picture list reordering commands start with the character ‘R’ followed by an optional ordered list of re-ordering commands.

RPLR-List0: R[–DIFF|+DIFF|L(LTI)]…[–DIFF|+DIFF|L(LTI)]
RPLR-List1: R[–DIFF|+DIFF|L(LTI)]…[–DIFF|+DIFF|L(LTI)]

The reference list re-ordering commands are specified in the following way.

–DIFF: specifies the reference picture list reordering command reordering_of_pic_nums_idc = 0, the associated syntax element abs_diff_pic_num_minus1 is set equal to the given decimal number DIFF.
+DIFF: specifies the reference picture list reordering command reordering_of_pic_nums_idc = 1, the associated syntax element abs_diff_pic_num_minus1 is set equal to the given decimal number DIFF.
L(LTI): specifies the reference picture list reordering command reordering_of_pic_nums_idc = 2, the associated syntax element long_term_pic_num is set equal to the given decimal number LTI. [Currently not supported by the encoder]

The reference picture list reordering command reordering_of_pic_nums_idc = 3 (end of list) is not specified, but automatically appended if the specified list of reordering commands is not empty. For B frames, two lists of reordering commands can be specified, the first list is related to reference picture list 0 and the second list is related to reference picture list 1. If reference picture list reordering commands should only be specified for reference picture list 1, this can be done by specifying an empty list for reference picture list 0: RR[–DIFF|+DIFF|L(LTI)]…[–DIFF|+DIFF|L(LTI)].

The list of memory management control operation (MMCO) commands starts with the character ‘M’ followed by an optional ordered list of MMCO commands.

**MMCO-List:** M[N(DIFF)|L(LTI)–|L(LTI):(DIFF)|L(NUM)$|E|L(LTI)+]…[…]

The MMCO commands are specified in the following way.

**N(DIFF):** specifies the memory management control operation command memory_management_control_operation = 1 (mark a short-term picture as “unused for reference”), the associated syntax element difference_of_pic_nums_minus1 is set equal to the given decimal number DIFF.

**L(LTI)–:** specifies the memory management control operation command memory_management_control_operation = 2 (mark a long-term picture as “unused for reference”), the associated syntax element long_term_pic_num is set equal to the given decimal number LTI. [Currently not supported by the encoder]

**L(LTI):(DIFF):** specifies the memory management control operation command memory_management_control_operation = 3 (assign a long-term index to a short-term picture), the associated syntax elements long_term_frame_idx and difference_of_pic_nums_minus1 are set equal to the given decimal numbers LTI and DIFF, respectively. [Currently not supported by the encoder]

**L(NUM)$:** specifies the memory management control operation command memory_management_control_operation = 4 (specify maximum long-term frame index), the associated syntax element max_long_term_frame_idx_plus1 is set equal to the given decimal number NUM. [Currently not supported by the encoder]

**E:** specifies the memory management control operation command memory_management_control_operation = 5 (mark all reference pictures as “unused for reference” and set the MaxLongTermFrameIdx variable to “no long-term frames indices”). [Currently not supported by the encoder]

**L(LTI)+:** For non-IDR pictures, it specifies the MMCO command memory_management_control_operation = 6 (assign a long-term frame index to the current decoded picture), the associated syntax element long_term_frame_idx is set equal to the given decimal number LTI. [Currently not supported by the encoder]

For IDR pictures, it sets the flag long_term_reference_flag equal to 1 and the given decimal number LTI is ignored. [Currently not supported by the encoder]
2.2.1.2.2 Examples for using the sequence structure string

In the following various examples are given for the usage of the sequence structure format string specified above.

2.2.1.2.2.1 Examples without MMCO and reordering commands

Example 1:
Format string: \( A0*n\{P0\} \)
Coding Types: IDR P P P P P P ...  
Stored as reference: 1 1 1 1 1 1 1 ...  
Coding Order: 0 1 2 3 4 5 6 ...

Example 2:
Format string: \( A0*n\{P2b0b1\} \)
Coding Types: IDR B B P B B P ...  
Stored as reference: 1 0 0 1 0 0 1 ...  
Coding Order: 0 2 3 1 5 6 4 ...

Example 3:
Format string: \( A0*2\{P3B1b0b2\} \)
Coding Types: IDR B B B P B B B P IDR B B P ...  
Stored as reference: 1 0 1 0 1 0 1 1 0 1 0 1 ...  
Coding Order: 0 3 2 4 1 7 6 8 5 9 12 11 13 10...

Example 4:
Format string: \( A0*2\{P1p0\}*n\{I0*2\{P1p0\}\} \)
Coding Types: IDR P P P P I P P P P I P P P P ...  
Stored as reference: 1 0 1 0 1 1 0 1 1 0 1 0 1 ...  
Coding Order: 0 2 1 4 3 5 7 6 9 8 10 12 11 14 13...

2.2.1.2.2.2 Example with MMCO commands

Example 5:
Reference frames: 5
Format string: \( A0P1ML2$L0+P2ML1:1P3ML0-N0P4P5P6MEP7P8P9 \)
Coding Types: IDR P P P P P P P P P IDR ...  
Stored as reference: 1 1 1 1 1 1 1 1 1 1 1 ...  
Coding Order: 0 1 2 3 4 5 6 7 8 9 10 ...

<table>
<thead>
<tr>
<th>Current Frame Index (coding order)</th>
<th>Ordered set of short-term reference pictures (specified by the frame indices) after decoding and storing the current frame</th>
<th>Ordered set of long-term reference pictures (specified by the frame indices) after decoding and storing the current frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 (1)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2 (2)</td>
<td>2</td>
<td>1 0</td>
</tr>
<tr>
<td>3 (3)</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4 (4)</td>
<td>4 3</td>
<td>0</td>
</tr>
</tbody>
</table>
2.2.1.2.2.3 Example with Reordering Commands

Example 6:
Reference frames: 4
Format string: A0P4B1B3b2R-0-0-0R-1+0-2
Coding Types: IDR B B B P IDR B B B P ...
Stored as reference: 1 1 0 1 1 1 1 0 1 1 ...
Coding Order: 0 2 4 3 1 5 7 9 8 6 ...

<table>
<thead>
<tr>
<th>Current Frame Index (coding order)</th>
<th>Reference index list 0 (specified by the frame indices); in brackets: list 0 without reordering if it is different</th>
<th>Reference index list 1 (specified by the frame indices); in brackets: list 1 without reordering if it is different</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4 (1)</td>
<td>0 4</td>
<td>4 0</td>
</tr>
<tr>
<td>1 (2)</td>
<td>0 4</td>
<td>4 0</td>
</tr>
<tr>
<td>3 (3)</td>
<td>1 0 4</td>
<td>4 1 0</td>
</tr>
<tr>
<td>2 (4)</td>
<td>3 1 4 0 (1 0 3 4)</td>
<td>1 3 0 4 (3 4 1 0)</td>
</tr>
<tr>
<td>5 (5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 (6)</td>
<td>5</td>
<td>9 5</td>
</tr>
<tr>
<td>6 (7)</td>
<td>5 9</td>
<td>9 5</td>
</tr>
<tr>
<td>8 (8)</td>
<td>6 5 9</td>
<td>9 6 5</td>
</tr>
<tr>
<td>7 (9)</td>
<td>8 6 9 5 (6 5 8 9)</td>
<td>6 8 5 9 (8 9 6 5)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

2.2.1.3 Command line options

Command line option are processed in the order they appear on the command line, previously specified parameters are overwritten. The following command line options are available in single-layer coding mode:

- **-bf (bitstream)**
  The parameter bitstream specifies the filename for the bit-stream to be generated.

- **-frms (frames)**
  The parameter frames specifies the number of frames of the input sequence to be encoded.

- **-pf (config)**
  The parameter config specifies the name of the config file to be used.

- **-h**
  Prints out a brief help on using the encoder.
### 2.2.2 Scalable coding mode

The encoder is run in scalable coding mode, when the configuration file does not contain the parameter *MVCMode* or when this parameter is set equal to 0. When running the encoder in scalable coding mode, one or more layer configuration files have to be specified inside the main configuration files. The parameters of the main configuration file are described in section 2.2.2.1, the parameters of the layer configuration files are described in section 2.2.2.2. Additional command line parameters are described in section 2.2.2.3.

#### 2.2.2.1 Main configuration file

All available parameters for the main configuration file together with a brief description are summarized in Example 13. Additional information about the configuration parameters, including default values, are given below.

**Example 13: Main encoder configuration file in scalable coding mode**

```plaintext
# JSVM Main Configuration File

#---------------------------- GENERAL ----------------------------------------
OutputFile              test.264  # Bitstream file
FrameRate               30.0      # Maximum frame rate [Hz]
MaxDelay                1200.0    # Maximum structural delay [ms]
FramesToBeEncoded       100       # Number of frames (at input frame rate)
NonRequiredEnable       0         # NonRequiredSEI enable (0:disable, 1:enable)
CgsSNRRefinement........0         # SNR refinement as 1: CGS (EI, EP, EB) 0: FGS (PR)

#---------------------------- CODING STRUCTURE ------------------------------
GOPSize                 2          # GOP Size (at maximum frame rate)
IntraPeriod             4          # Intra Period
NumberReferenceFrames   1          # Number of reference pictures
BaseLayerMode           1          # Base layer mode (0 : AVC with larger DPB,
                                  #   1:AVC compatible, 2:AVC w subseq SEI)

#---------------------------- 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

#---------------------------- LAYER DEFINITION -----------------------------
NumLayers               2          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
LayerCfg                layer1.cfg # Layer configuration file

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

#---------------------------- LOSS-AWARE RDO -------------------------------
LARDO                   0          # Loss-aware RDO (0:disable, 1:enable)

#---------------------------- OTHER PARAMETERS -----------------------------
SuffixUnitEnable        1          # Add suffix unit (0: off, 1: on) shall always be on
                                  # in SVC contexts (i.e. when there are
                                  # FGS/CGS/spatial enhancement layers)
```


<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMC0BaseEnable</td>
<td>1</td>
<td># MMC0 for base representation (0: off, 1: on)</td>
</tr>
<tr>
<td>NestingSEI</td>
<td>1</td>
<td># Nesting SEI message (1: enable, 0: disable)</td>
</tr>
<tr>
<td>SceneInfo</td>
<td>1</td>
<td># scene info SEI message (1: enable, 0: disable)</td>
</tr>
</tbody>
</table>

**OutputFile**

String, default: test.264  
Specifies the filename for the bit-stream to be encoded.

**FrameRate**

Double, default: 60.0  
Specifies the maximum frame rate of all input sequences or a multiple thereof. The parameter *FramesToBeEncoded* specifies the number of frames that should be encoded at the input frame rate *FrameRate*. The parameter *FrameRate* is additionally required for determining the actual coding structure of a layer. The basic coding structure (GOP size) for a layer is determined by the parameters *FrameRate* and *GOPSize* of the main configuration files and the parameters *FrameRateIn* and *FrameRateOut* of the layer configuration files (cp. section 2.2.2.2). See also *GOPSize*.

**MaxDelay**

Double, default: 1200.0  
Specifies the maximum allowed structural delay in milliseconds (ms). In order to not exceed this maximum delay the coding structure might be adapted by restricting the motion-compensated prediction to reference pictures from the past for several or all pictures. In general the coding efficiency is decreased by forcing a maximum structural delay.

**FramesToBeEncoded**

Unsigned Int, default: 1  
Specifies the number of frames of the input sequence to be encoded. The number of frames to be encoded is specified at the frame rate given by *FrameRate*. Thus, depending on the value of *FrameRateOut* in the layer configuration file(s), the number of actual encoded frame might be smaller.

**NonRequiredEnable**

Flag (0 or 1), default: 0  
Specifies whether non-required picture SEI messages are included in the generated bit-stream. If *NonRequiredEnable* is equal to 1, non-required picture SEI messages are included in the bit-stream; otherwise, non-required picture SEI messages are not included in the bit-stream.

**CgsSnrRefinement**

Flag (0 or 1), default: 0  
Specifies whether SNR enhancements are coded using EI, EP or EB slices (flag set to a value of 1) or coded using PR slices (flag set to a value of 0).

**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 a key picture, which is generally coded as P picture, and several hierarchically coded B pictures that are located between the key pictures. The parameter *GOPSize* must be equal to a power of 2. The GOP size is specified at the frame rate given by *FrameRate*. Thus, depending on the value of *FrameRateOut* in the layer configuration file, the actual GOP size for a layer might be smaller. For example, if *FrameRate* is equal to 30, *GOPSize* is equal to 16, and *FrameRateOut* is equal to 7.5, the actual GOP size that is employed for encoding the specific layer is equal to 4. The GOP size of all layers is selected in a way that the key pictures for all layers are temporally aligned. Hence, depending on the parameters *FrameRateOut* in the layer configuration files, the allowed range for *GOPSize* might be restricted. The maximum allowed value is 64.

**IntraPeriod**

Unsigned Int, default: $2^{32} - 1$ (equal to $-1$)
Specifies the intra period for the encoded video sequence. When \textit{IntraPeriod} is equal to \(-1\ (2^{32}-1)\), only the very first picture is intra coded. Otherwise, every \textit{IntraPeriod} picture (at the frame rate \textit{FrameRate}) is intra-coded. The parameter \textit{IntraPeriod} shall be equal to \(-1\) or equal to a multiple of \textit{GOPSize}.

\textbf{NumberReferenceFrames}

\textit{Unsigned Int, default: 1}

Specifies the maximum number of active entries for the 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.

\textbf{BaseLayerMode}

\textit{Unsigned Int, default: 3}

Specifies the AVC compatibility of the base layer (LayerId equal to 0). The following values are supported:

- 0 – AVC compatible base layer with larger DPB size,
- 1 – AVC compatible base layer,
- 2 – AVC compatible base layer with sub-sequence SEI messages for supporting temporal scalability.

This parameter shall be always present since the default value is invalid. Note that an extraction of temporal sub-layers is only possible when \textit{BaseLayerMode} is equal to 0 or 2.

\textbf{NumLayers}

\textit{Unsigned Int, default: 1}

Specifies the number of layers. A layer represents a spatial layer or a coarse-grain SNR scalable layer. Note that the number temporal layers is specified by the parameter \textit{GOPSize}, and that fine-grain SNR scalable layers are specified by the parameter \textit{NumFGSLayers} in the layer configuration files (a “layer” can contain several FGS layers). The parameter \textit{NumLayers} shall be in the range of 1 to 8, inclusive. For each layer a layer configuration file shall be specified by using the parameter \textit{LayerCfg}.

\textbf{LayerCfg}

\textit{String, no default value}

Specifies the filename of a layer configuration file. The main configuration file shall contain exactly \textit{NumLayers} occurrences of the parameter \textit{LayerCfg}. Each of these specifies the layer configuration file for a specific layer. The first occurrence of the parameter \textit{LayerCfg} specifies the layer configuration file for the base layer (LayerId 0), the next occurrence specified the layer configuration file for the next layer (LayerId 1), etc.

\textbf{LARDO}

\textit{Flag (0 or 1), default: 0}

Specifies whether loss-aware rate-distortion optimized macroblock mode decision is used. This parameter is evaluated when the parameter \textit{NumFGSLayers} in the layer configuration files is equal to 0 and the parameter \textit{ClosedLoop} in the layer configuration files is not equal to 0.

\textbf{SearchMode, SearchFuncFullPel, SearchFuncSubPel, SearchRange, BiPredIter, IterSearchRange, LoopFilterDisable, LoopFilterAlphaC0Offset, LoopFilterBetaOffset, WeightedPrediction, WeightedBiPrediction}

These parameters are described in section 2.2.1.1.

\textbf{SuffixUnitEnable}

\textit{Bool, default: 1}

Specifies whether to add a suffix unit after each AVC coded slice NAL unit. When this parameter is equal to 0, no suffix NAL unit is added in the bitstream. When this parameter is 1, a suffix NAL unit is added
after each AVC code slice NAL unit. This parameter shall always be on in SVC contexts (i.e. when there are FGS/CGS/spatial enhancement layers).

**MMCOBaseEnable**

*Bool, default: 1*

Specifies whether MMCO is used for the management of the base representation of key pictures. When this parameter is equal to 0, no MMCO for base representations is included in the bitstream and the decoder uses sliding window to manage the base representations. When this parameter is equal to 1, MMCO for base representation is included in the bitstream.

**NestingSEI**

*Bool, default: 0*

Specifies whether scalable nesting SEI message is included in the bitstream. When this parameter is equal to 0, no scalable nesting SEI message is included. When this parameter is equal to 1, scalable nesting SEI messages containing AVC SEI messages are included in the bitstream.

**SceneInfo**

*Bool, default: 0*

Specifies whether scene information SEI message is included in the bitstream. When this parameter is equal to 0, no scene information SEI message is included. When this parameter is equal to 1, scalable nesting SEI messages containing AVC scene information SEI messages are included in the bitstream.

### 2.2.2.2 Layer configuration file

All available parameters for the layer configuration file together with a brief description are summarized in Example 14. Additional information about the configuration parameters, including default values, are given below.

*Example 14: Layer configuration for encoding in scalable coding mode*

```bash
# JSVM Layer Configuration File

#---------------------- INPUT / OUTPUT ------------------------
SourceWidth          176           # Input frame width
SourceHeight         144           # Input frame height
FrameRateIn          15            # Input frame rate [Hz]
FrameRateOut         15            # Output frame rate [Hz]
InputFile            orig.yuv      # Input file
ReconFile            rec_layer.yuv # Reconstructed file
SymbolMode           0             # 0=CAVLC, 1=CABAC
IDRPeriod            0             # IDR period (should be (GOP size*N))
UseAdaptiveGOP       0

#---------------------- CODING -------------------------------
ClosedLoop           2             # Closed-loop control (0,1:at highest
                                     #   rate, 2: at lowest and highest rate)
FRExt                1             # 8x8 trafo (0:off, 1:on)
MaxDeltaQP           1             # Max. absolute delta QP
QP                   35.0          # Quantization parameters
NumFGSLayers         0.0           # Number of FGS layers (1 layer -> DQP = 6)
FGSMotion            0             # motion ref in FGS slices (0:off, 1:on)
BaseLayerId          1             # Layerd ID for the base layer
ForceReOrdering      1             # Force RPLR commands (0:off, 1:on)
EncSIPFile           layer1.dat    # SIP decision file

#---------------------- CONTROL -------------------------------
MeQPLP               30.00         # QP for mot. est. / mode decision (key pics)
MeQP0                30.00         # QP for mot. est. / mode decision (stage 0)
MeQP1                30.00         # QP for mot. est. / mode decision (stage 1)
MeQP2                30.00         # QP for mot. est. / mode decision (stage 2)
MeQP3                30.00         # QP for mot. est. / mode decision (stage 3)
MeQP4                30.00         # QP for mot. est. / mode decision (stage 4)
MeQP5                30.00         # QP for mot. est. / mode decision (stage 5)
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterLayerPred</td>
<td>0</td>
<td>Inter-layer Pred. (0: no, 1: yes, 2: adap.)</td>
</tr>
<tr>
<td>BaseQuality</td>
<td>3</td>
<td>Base quality level (0, 1, 2, 3: all)</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>MotionInfoMode</td>
<td>0</td>
<td>Motion Info Mode (0:no, 1:load, 2: save)</td>
</tr>
<tr>
<td>MotionInfoFile</td>
<td>layer.mot</td>
<td>Motion Info File</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>UseESS</td>
<td>2</td>
<td>ESS mode</td>
</tr>
<tr>
<td>ESSPicParamFile</td>
<td>crop.txt</td>
<td>Picture level cropping parameters</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>UseESS</td>
<td>2</td>
<td>ESS mode</td>
</tr>
<tr>
<td>ESSPicParamFile</td>
<td>crop.txt</td>
<td>Picture level cropping parameters</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>UseRedundantSlc</td>
<td>1</td>
<td>UseRedundantSlice, 0: not used, 1: one redundant slice used for each slice</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>PLR</td>
<td>3</td>
<td>Packet Loss Rate</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
**SourceWidth**

*Unsigned Int, default: 176*

Specifies the width of the input images in luma samples. `SourceWidth` shall be non-zero and a multiple of 16.

**SourceHeight**

*Unsigned Int, default: 352*

Specifies the height of the input images in luma samples. `SourceHeight` shall be non-zero and a multiple of 16.

**FrameRateIn**

*Double, default: 30*

Specifies the frame rate of the input sequence. The parameter `FrameRate` in the main configuration file shall be a multiple of `FrameRateIn`, and `FrameRateIn` shall be a multiple of `FrameRateOut`.

**FrameRateOut**

*Double, default: 30*

Specifies the output frame rate for the current layer. The parameter `FrameRate` in the main configuration file shall be a multiple of `FrameRateIn`, and `FrameRateIn` shall be a multiple of `FrameRateOut`. When `FrameRateOut` is not equal to `FrameRateIn`, every (\(\frac{FrameRateIn}{FrameRateOut}\)) frame of the input sequence is skipped. The actual GOP size that is used for encoding a layer is determined by the parameters `FrameRate`, `GOPSize`, and `FrameRateOut`. The actual number of frames that are coded for a layer is determined by the parameters `FrameRate` and `FrameRateOut`.

**InputFile**

*String, default: “test.yuv”*

Specifies the filename of the original raw video sequence for the layer.

**ReconFile**

*String, default: “rec.yuv”*

Specifies the filename of the coded and reconstructed input sequence for the layer. This sequence is provided for debugging purposes.

**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.

**IDRPeriod**

*Int, default: 0*

Specifies the period of IDR pictures inside the layer. The set of possible values is given by \((GOPSize*N)\) with \(N\) being 0, 1, ...

**ClosedLoop**

*Unsigned Int, default: 0*

Specifies whether the encoder is operated in open- or closed-loop mode. The following values are supported:

- 0, 1 – One-point closed-loop encoder control
- 2 – Two-point closed-loop encoder control

When `ClosedLoop` is equal to 0 or 1, the prediction loop is closed at the highest quality point. When `ClosedLoop` is equal to 2, the prediction loop is closed at two points, the lowest and highest quality point. When no FGS layers are present, identical results are obtained with `ClosedLoop` equal to 0, 1 or 2. When FGS layers are encoded and `ClosedLoop` is set equal to 2, the rate-distortion efficiency at the lowest supported bit-rate is usually increased while the rate-distortion efficiency at the highest bit-rate is usually decreased in comparison to the mode with `ClosedLoop` equal to 0 or 1.
FRExt
Flag (0 or 1), default: 0
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.

MaxDeltaQP
Unsigned Int, default: 1
Specifies the maximum absolute difference of macroblock quantization parameters from the picture quantization parameter.

QP
Double, default: 32.0
Specifies the basis quantization parameter for encoding the current layer. The actual quantization parameters for encoding a specific frame are determined depending on this parameter QP and the position of the frame inside the group of pictures. Note that FGS layers are encoded with a smaller quantization parameter; for each FGS layer the quantization parameter is decreased by a value of 6.

NumFGSLayers
Double, default: 0
Specifies the number of FGS layers to be encoded inside the current spatial or coarse-grain SNR layer. NumFGSLayers shall be in the range of 0 to 3, inclusive. When a fractional number is specified, the last FGS layer is truncated after the corresponding percentage of coding symbols. This truncation ratio is not identical to the bit-rate ratio. Each FGS layer corresponds to a decrease of the quantization parameter by a value of 6.

FGSMotion
Flag (0 or 1), default: 0
Specified whether motion refinements are used in FGS slices. If FGSMotion is equal to 0, only transform coefficients are refined in FGS slices. If FGSMotion is equal to 1, it is also possible to transmit new motion data for macroblocks in FGS slices; whether a motion refinement is transmitted for a macroblock is determined via rate-distortion optimization.

BaseLayerId
Unsigned Int, default: next lower layer when present
Specifies that layer that shall be used for inter-layer prediction. When BaseLayerId is not present, the next lower layer (specified by the occurrence order in the main configuration file) is considered to represent the base layer for inter-layer prediction. Otherwise, the layer that is employed for inter-layer prediction is specified by ist LayerId. The LayerId is determined by the occurrence order in the main configuration file. See LayerCfg.

ForceReOrdering
Flag (0 or 1), default: 0
Specifies whether reference picture list reordering commands are encoded for every entry of the reference picture lists. When ForceReOrdering is equal to 0, only required reference picture list reordering commands are transmitted. When ForceReOrdering is equal to 1, a reordering command is transmitted for each entry of the reference picture lists. By forcing reordering commands, the ability to detect missing pictures is increased.

EncSIPFile
String, default: “”
Specifies the name of the SIP decision file. This file records POCs of this layer’s frames which don’t use inter-layer prediction. When EncSIPFile is not present, the encoder doesn’t use SIP strategy on this layer.

MeQPLP
Double, default: QP
Specifies the Lagrangian parameter that is used for motion estimation and mode decision of key pictures. The Lagrangian parameter for mode decision is determined by \( L = 0.85 \times (2 \wedge (MeQPLP / 3)) \), the Lagrangian parameter for motion estimation is additionally dependent on the distortion measure that is used for the motion search. When this parameter is not specified, it is inferred to be equal to the parameter QP.
**MeQPX** (with X being replaced by 0, 1, 2, 3, 4, or 5)

*Double, default: 32.0*

Specifies the Lagrangian parameter that is used for motion estimation and mode decision of the temporal layer X. The Lagrangian parameter for mode decision is determined by \( L = 0.85 \times (2^{\text{MeQPX}/3}) \), the Lagrangian parameter for motion estimation is additionally dependent on the distortion measure that is used for the motion search.

**InterLayerPred**

*Unsigned Int, default: 0*

Specifies the usage of inter-layer prediction. The following values are supported:

- 0 – The layer does not use inter-layer prediction
- 1 – Inter-layer prediction is always used
- 2 – Inter-layer prediction is used in a macroblock adaptive way

InterLayerPred shall always be equal to 0 for the base layer (LayerId 0). With mode 1, the inter-layer prediction is always enabled for all macroblocks of a pictures as long as a corresponding base layer picture exists. That mean especially that the motion vectors are always copied from the base layer and that the residual data of the base layer are always used for prediction of the current residuals. When InterLayerPred is equal to 2, the inter-layer prediction tools are arbitrarily selected via a rate-distortion optimization framework. For scalable enhancement layers, the best coding efficiency is obtained when InterLayerPred is set equal to 2.

**BaseQuality**

*Unsigned Int, default: 3*

Specifies which quality layer (FGS layer) of the base layer is used for inter-layer prediction. Since up to 3 FGS layers can be present for a spatial or coarse-grain layer, BaseQuality shall be in the range of 0 to 3, inclusive. A value of 0 for BaseQuality specifies that only the base representation and none of the FGS layers are employed for inter-layer prediction; a value of 1 specifies that the data of the first FGS layer are used for inter-layer prediction; a value of 2 specifies that the data of the second FGS layer are used for inter-layer prediction, etc. When the number of actually present FGS layers is less than the specified value of BaseQuality, all available FGS layers are employed for inter-layer prediction. Especially, by specifying the default value of 3, all base layer data are used in any case. It should be noted that by specifying a value less than 3 for BaseQuality.

**MotionInfoMode**

*Unsigned Int, default: 0*

Specifies whether motion data are loaded from or saved to a file. The following values are supported:

- 0 – motion data are neither saved nor loaded
- 1 – motion data are loaded from the file specified by MotionInfoFile
- 2 – motion data are saved to the file specified by MotionInfoFile

Note that only motion data for non-key picture are loaded from or saved to a motion data file. When closed-loop coding is employed and the quantization parameters are changed, the loaded motion data for macroblocks coded in direct mode could be inappropriate. Special care should be taken when using this possibility to load motion data.

**MotionInfoFile**

*String, default: “test.mot”*

Specifies the filename of the motion data while that is used for loading or storing motion data of non-key pictures as indicated by the parameter MotionInfoMode.

**UseESS**

*Int, default: 0*

Specifies whether Extended Spatial Scalability (ESS) should be used. ESS enables a generalized relation between successive spatial layers. A picture of a lower spatial layer may represent a cropped area of the higher resolution picture and the relation between successive spatial layers does not need to be dyadic. Geometrical parameters defining the cropping window and the down-sampling ratio can either be defined at the sequence level, or evolve at the picture level. The following values are supported:

- 0 – no ESS
- 1 – Sequence level ESS
- 2 – Picture level ESS

**ESSPicParamFile**
String, default: “ess.dat”
Specifies the filename containing the *cropping window* parameters. This option is only required when the UseESS option is set to 2 (Picture level ESS). Each line of the file is made up of four integers separated by commas as following: “x_orig, y_orig, crop_width, crop_height”. Where x_orig and y_orig represent the coordinates of the upper left corner of the *cropping window* and crop_width and crop_height represent respectively its width and height. The *cropping window* parameters are given in the high layer domain, indeed it (somehow) represents the low layer upscaled. In case, the file contains less lines than the number of pictures in the output sequence, the cropping parameters of the last line will be used for the following pictures.

**ESSCropWidth**
**Int, default: 0**
Specifies the width of the *cropping window*. The UseESS option should be set to 1 or 2.

**ESSCropHeight**
**Int, default: 0**
Specifies the height of the *cropping window*. The UseESS option should be set to 1 or 2.

**ESSOriginX**
**Int, default: 0**
Specifies the X-position of the upper left corner of the *cropping window*. The UseESS option should be set to 1.

**ESSOriginY**
**Int, default: 0**
Specifies the Y-position of the upper left corner of the *cropping window*. The UseESS option should be set to 1.

**ESSChromaPhaseX**
**Int, default: -1**
Specifies the high layer (current layer) chroma component phase shift in dimension X in comparison to the luma component in dimension X. Values are given in quarter luma samples (in range [-1;1]). The allowed value for this parameter should be in the range of -1 to 1, inclusive.

**ESSChromaPhaseY**
**Int, default: 0**
Specifies the high layer (current layer) chroma component phase shift in dimension X in comparison to the luma component in dimension Y. Values are given in quarter luma samples (in range [-1;1]). The allowed value for this parameter should be in the range of -1 to 1, inclusive.

**ESSBaseChromaPhaseX**
**Int, default: -1**
Specifies the base layer (lower than the current layer) chroma component phase shift in dimension X in comparison to the luma component in dimension X. Values are given in quarter luma samples (in range [-1;1]). The allowed value for this parameter should be in the range of -1 to 1, inclusive.

**ESSBaseChromaPhaseY**
**Int, default: 0**
Specifies the base layer (lower than the current layer) chroma component phase shift in dimension X in comparison to the luma component in dimension Y. Values are given in quarter luma samples (in range [-1;1]). The allowed value for this parameter should be in the range of -1 to 1, inclusive.

**EnhRefME**
**Double, default: 0.0**
Specifies how the reference frame that is used for motion estimation is formed. Basically, it specifies a weighting factor between base layer reference frame and enhancement layer reference frame. The reference frame used for motion estimation should be: base*(1-a)+enh*a with a being the parameter EnhRefME. The allowed value for this parameter should be in the range of 0 to 1, inclusive. Usually for AR-FGS, a value of 0.5 is selected.

**WeightZeroBlock**
**Unsigned Int, default: 32**
Specifies the weighting factor for zero blocks that is used for AR-FGS of key pictures.

WeightZeroCoeff
Unsigned Int, default: 32
Specifies the weighting factor for zero coefficients that is used for AR-FGS of key pictures.

FgsEncStructure
Unsigned Int, default: 1
Specifies the FGS syntax structure that is used for AR-FGS of key pictures. If FGSEncStructure is equal to 0, the standard FGS syntax is used. If FGSEncStructure is equal to 1, an optimized FGS syntax structure is used.

FGSVectorMode
Flag (0 or 1), default: 0
Specifies the vector scanning mode of FGS coefficients. A value of 0 specifies that grouping size is specified, 1 specifies that a vector is specified. Grouping size indicates the number of FGS coefficients coded for each block at each cycle. The vector specifies the position to reach for each block at each cycle. Vector is specified for 4x4 blocks, the vector for 8x8 blocks is then inferred.

FGSGroupingSize
Unsigned Int, default: 1
Specifies the size of the group of the FGS coefficients coded for each block at each cycle.

FGSVectorX (with X being replaced by 0–15)
Unsigned Int, default: 0
Specifies the X-th position of the vector. The last position shall be equal to 15.

SliceMode
Unsigned Int, default: 0
Sets slice mode: 0-disabled, 1-fixed number of MB per slice, 2-fixed number of bytes per slice, 3-use callback, 4-rectangular grid slice for IROI. [Ed. Note(Heiko): The description needs to be improved. It looks like it is simply copied from the JM manual. But as far as I know the described mode are not supported.]

SliceArgument
Unsigned Int, default: 50
Slice arguments for modes 1 (number of MBs), 2 (bytes) and 4 (grid slice height in MBs). [Ed. Note(Heiko): The description needs to be improved. It looks like it is simply copied from the JM manual. But as far as I know the described behaviour is not supported.]

NumSlicGrpMns1
Unsigned Int, default: 0
Specifies the number of slice groups. A value of 0 specified one slice group, a value of 1 specifies 2 slice groups, etc.

SlcGrpMapType
Unsigned Int, default: 2
Specifies the slice group map type: The following values are supported:

0 – Interleaved mode
1 – Dispersed mode
2 – Foreground with left-over
3 – Box-out
4 – Raster scan
5 – Wipe
6 – Explicit mode, slice_group_id is read from the file specified by SlcGrpCfgFileNm

SlcGrpChgDrFlag
Unsigned Int, default: 0
Sets slice_group_change_direction_flag. The following values are supported:

0 – Box-out clockwise, raster scan or wipe right
1 – Box-out counter clockwise, reverse raster scan or wipe left

SlcGrpChgRtMns1
Unsigned Int, default: 85
Specifies the value of the syntax element slice_group_change_rate_minus1.

SlcGrpCfgFileNm
String, default: “sgcfg.cfg”
Slice configuration file used for slice group map types 0, 2, and 6.
If NumSlcGrpMns1>0 and SlcGrpMapType=0 and SliceMode=4, the sum of runlength of all slice
groups must equal to (SourceWidth/16). IROI coding is enabled and IROI slice division information is
coded.

UseRedundantSlc
Unsigned Int, default: 0
Enables the use of redundant slices. Currently supports one redundant slice per slice except FGS coded
one. The current redundant slice tool does not support multiple slices per picture.

PLR
Unsigned Int, default: 0
Specifies the packet loss rate for the current layer. This parameter is used by the loss-aware rate-distortion
optimization, see LARDO.

NumROI
Unsigned Int, default: 0
Specifies the number of ROIs in the current layer.

ROICfgFileNm
String, default “roiconf.cfg”
ROI configuration file. If NumROI > 0, roi_id, slice group id, scalability layer id for each ROI is
described.

2.2.2.3 Command line

2.2.2.4 options

Command line option are processed in the order they appear on the command line, previously
specified parameters are overwritten. The following command line options are available in scalable
coding mode:

-anafgs (layer) (numFGSLayers) (FGSDataFile)
  This option can be used in connection with the option –encfgs to generate bit-streams that contain FGS
  layers with a given maximum bit-rate. This option specifies that numFGSLayers FGS layers are encoded
  for the CGS or spatial layer with LayerId layer, and that information about the coded FGS packet sizes
  are stored in the file FGSDataFile. For rate-distortion efficient coding the number of FGS layer specified
  by numFGSLayers should be chosen in a way that the resulting bit-rate is close to the target bit-rate that is
  specified in a subsequent call using the option –encfgs.

-encfgs (layer) (bitRate) (FGSDataFile)
  This option can be used in connection with the option –anafgs to generate bit-streams that contain FGS
  layers with a given maximum bit-rate. This option specifies that the resulting bit-stream for the layer
  specified by layer should be equal to bitRate, with bitRate being the bit-rate in kbit/s. For adjusting the
  bit-rate the data about FGS packet size that are stored in the file FGSDataFile are used. The file
  FGSDataFile has to be generated in advance by calling the encoder with the option –anafgs.

-bf (bitstream)
  The parameter bitstream specifies the filename for the bit-stream to be generated.

-vlc
  Sets the parameter SymbolMode equal to 0 for all layers.
-cabac
Sets the parameter SymbolMode equal to 1 for all layers.

-numl (numLayers)
Specifies the number of layers for the encoder call. The parameter numLayers shall not exceed the value of parameter NumLayers in the main configuration file.

-org (layer) (original)
The parameter original overwrites the parameter InputFile of the layer configuration file for the layer given by layer.

-rec (layer) (reconstructed)
The parameter reconstructed overwrites the parameter ReconFile of the layer configuration file for the layer given by layer.

-ec (layer) (ecmode)
The parameter ecmode overwrites the parameter SymbolMode of the layer configuration file for the layer given by layer.

-rqp (layer) (rQP)
Specifies the basis quantization parameter rQP to be used for quantization of the layer specified by layer. Overwrites the parameter QP of the corresponding layer configuration file.

-mqp (layer) (stage) (mQP)
Specifies the Lagrangian parameter for the temporal level stage of the layer layer. Overwrites the parameters MeQPX and MeQPLP of the corresponding layer configuration file with X being equal to stage. The Lagrangian parameter is specified by a quantization parameter mQP as specified in the description of the parameters MeQPX and MeQPLP.

-lqp (layer) (lQP)
Specifies the basis quantization parameter and the Lagrangian parameters for the layer given by layer. The option --lqp X lQP is identical to the following list of options --rqp and --mqp:
--rqp X lQP --mqp X 0 lQP --mqp X 1 lQP --mqp X 2 lQP --mqp X 3 lQP --mqp X 4 lQP --mqp X 5 lQP

-ilpred (layer) (mode)
The parameter mode overwrites the parameter InterLayerPred of the layer configuration file for the layer given by layer.

-mfile (layer) (mode) (file)
The parameter mode overwrites the parameter MotionInfoMode of the layer configuration file for the layer given by layer. The parameter file overwrites the parameter MotionInfoFile of the layer configuration file for the layer given by layer.

-frms (frames)
The parameter frames specifies the number of frames of the input sequence to be encoded. The number of frames to be encoded is specified at the frame rate given by the parameter FrameRate in the main configuration file.

-bcip
This option specifies that the base layer (LayerId equal to 0) is encoded using constrained intra prediction. When a multi-layer sequence is encoded successively by starting with the base layer and adding one enhancement layer in each further encoder call (e.g. to match given bit-rates), this option shall be specified for the encoder call, in which only the base is encoded.

-cl (layer) (cl)
The parameter cl overwrites the parameter ClosedLoop of the layer configuration file for the layer given by layer.

-ref (enhRef)
The parameter enhRef overwrites the parameter EnhRefME in the layer configuration files for all layers. [Ed. (Heiko) Double-check whether “all layers” is really correct]

-ar (wZBlock) (wZCoeff)
The parameters $wZBlock$ and $wZCoeff$ overwrite the parameters $WeightZeroBlock$ and $WeightZeroCoeff$, respectively, in the layer configuration files for all layers. \footnote{	ext{Ed. (Heiko) Double-check whether “all layers” is really correct}}

-**fs** (encStruct)

The parameter $encStruct$ overwrites the parameters $FgsEncStructure$ in all layer configuration files. \footnote{	ext{Ed. (Heiko) Double-check whether “all layers” is really correct}}

-**pf** (config)

The parameter $config$ specifies the name of the main configuration file to be used.

-**ds** (layer) (dsrate)

The parameter $dsrate$ specifies the bit-rate of the non-discardable FGS part for the layer given by $layer$. All FGS data that exceed the specified bit-rate $dsrate$ build a so-called discardable sub-stream. This means that for inter-layer prediction of a higher layer, the corresponding data in the discardable sub-stream are not employed. These data are only used when the layer $layer$ is decoded. The bit-rate $dsrate$ is specified in kbit/s.

-**anaags** (AGSmode)

The parameter $AGSMode$ overwrites the parameter $AGSModeDecision$ of the main configuration file. \footnote{	ext{The AGS feature does not work correctly in the current software}}

-**anasip** (layer) (mode) (file)

This option intends to tell the encoder to output the bits information used by the SIPAnalyser. The parameter $mode$ specifies whether frames on the layer specified by the parameter $layer$ can use inter-layer prediction or not. If $mode$ is equal to 0, the inter-layer prediction on all the frames of this layer is retained. Otherwise, the inter-layer prediction is all forbidden. The parameter $file$ specifies the output filename. If --anasip is not present, the encoder doesn’t output the bits information used by the SIPAnalyser.

-**encsip** (layer) (file)

The parameter $file$ overwrites the parameter $EncSIPFile$ of the layer configuration file for the layer given by $layer$.

-**h**

Prints out a brief help on using the encoder.

2.3 Decoder “H264AVCDecoderLibTestStatic”

The decoder call is illustrated in Example 15. The parameter $str$ specifies the filename of the bit-stream to be decoded, and the parameter $rec$ specifies the filename for the reconstructed video sequence.

**Example 15: Using the decoder**

```
H264AVCDecoderLibTestStatic <str> <rec> [-ec <ec>][<maxPOCDiff>]
```

- **str**: bit-stream file (input)
- **rec**: reconstructed video sequence (output)
- **ec**: error concealment method (1-3)
- **maxPOCDiff**: maximum difference of Picture Order Counts of successive output frames

The optional parameter $maxPOCDiff$ specifies a maximum difference of Picture Order Counts of successive output frames. When the actual difference of the Picture Order Count values of successive output frames exceeds this maximum value, the last decoded picture is repeated. This option can be used for decoding bit-streams with packet/frame losses in a way that the number of output frames is identical to the number of originally encoded frames. The value of $maxPOCDiff$ shall be greater than 0. When this parameter is not present, $maxPOCDiff$ is inferred to be equal to infinity.

The parameter $ec$ specifies the error concealment method. The following values are supported:

1 – all macroblock are assumed to be coded using BL Skip
2 – frame copy
3 – all macroblocks are assumed to be coded in direct mode

The default value for ec is 0. If the value is equal to 0, we do not perform any error concealment and no packet loss detection. For the bit-streams which are not supported by the current packet loss detection, uiErrorConceal equals to 0 means that it will perform as JSVM without error concealment.

[Ed. (HS): There was also a “usage example” provided by the proponent. Since I did not understand what it say, I did not included it. For completeness, it is given below:

The whole decoding is modified by decoding multiple bit-streams being wrapped together. By modifying the nCount in the main function of H264AVCDecoderLibTest. A end_of_stream packet is used among the packets.
]

2.4 Bit-stream extractor “BitStreamExtractorStatic”

The bit-stream extractor can be used to extract sub-streams of an AVC or SVC stream. The sub-streams represent streams with a reduced spatial and/or temporal resolution and/or a reduced bit-rate. The usage of the bit-stream extractor is illustrated in Example 16. A printout of the options that are provided by the bit-stream extractor can be obtained by calling the extractor without any command line parameter.

Example 16: Using the bit-stream extractor

<table>
<thead>
<tr>
<th>BitStreamExtractorStatic [-pt trace] &lt;in&gt;</th>
</tr>
</thead>
</table>

options:
- pt trace  -> generates a packet trace file from the given stream
- sl SL     -> extract the layer with layer id = SL and the dependent lower layers
- l L       -> extract all layers with dependency_id <= L
- t T       -> extract all layers with temporal_level <= T
- f F       -> extract all layers with quality_level <= F
- b B       -> extract a layer (possibly truncated) with the target bitrate = B
- e AxB@C:D -> extract a layer (possibly truncated) with
  A: frame width [luma samples]
  B: frame height [luma samples]
  C: frame rate [Hz]
  D: bit rate [kbit/s]
- ql        -> information about quality layers are used during extraction
- qlord     -> ordered/quality layer extraction
- sip       -> extract using selective inter-layer prediction strategy.
- suf       -> suffixUnit is used as auxiliary information during extraction when
  -sip is present and the base layer is AVC compatible.
- et trace  -> uses a (modified) packet trace file for bit-stream extraction
- ro roi_id -> extract roi which is identified by roi_id
- keepf     -> use with -l and -f options: extract all included layers of the layer
  L specified with -l and all quality levels below quality level F
  specified with -f of the layer L

Options "-l", "-t" and "-f" can be used in combination with each other.
The options "-ql" or "-qlord" can be used in connection with the option "-e".
The option "-sip" and "-suf" must be used in connection with the option "-e".
The option "-r" must be used in connection with the option "-e".
Other options can only be used separately.

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

When the extractor is called with a single parameter input.svc specifying the input bit-stream, information about the contained scalable layers are displayed as illustrated in Example 17. The DTQ information represents a triple of the values (dependency_id, temporal_level, quality_level).
Example 17: Using the bit-stream extractor for displaying information about a bit-stream

```
> BitStreamExtractorStatic input.svc

<table>
<thead>
<tr>
<th>Layer</th>
<th>Resolution</th>
<th>Framerate</th>
<th>Bitrate</th>
<th>DTQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>176x144</td>
<td>3.7500</td>
<td>106.00</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>176x144</td>
<td>3.7500</td>
<td>206.00</td>
<td>(0,0,1)</td>
</tr>
<tr>
<td>2</td>
<td>176x144</td>
<td>3.7500</td>
<td>341.00</td>
<td>(0,0,2)</td>
</tr>
<tr>
<td>3</td>
<td>176x144</td>
<td>7.5000</td>
<td>430.00</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>4</td>
<td>176x144</td>
<td>7.5000</td>
<td>454.00</td>
<td>(0,1,1)</td>
</tr>
<tr>
<td>5</td>
<td>176x144</td>
<td>7.5000</td>
<td>508.00</td>
<td>(0,1,2)</td>
</tr>
<tr>
<td>6</td>
<td>176x144</td>
<td>15.0000</td>
<td>594.00</td>
<td>(0,2,0)</td>
</tr>
<tr>
<td>7</td>
<td>176x144</td>
<td>15.0000</td>
<td>631.00</td>
<td>(0,2,1)</td>
</tr>
<tr>
<td>8</td>
<td>176x144</td>
<td>15.0000</td>
<td>721.00</td>
<td>(0,2,2)</td>
</tr>
<tr>
<td>9</td>
<td>352x288</td>
<td>3.7500</td>
<td>666.00</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>10</td>
<td>352x288</td>
<td>3.7500</td>
<td>1010.00</td>
<td>(1,0,1)</td>
</tr>
<tr>
<td>11</td>
<td>352x288</td>
<td>7.5000</td>
<td>1364.00</td>
<td>(1,1,0)</td>
</tr>
<tr>
<td>12</td>
<td>352x288</td>
<td>7.5000</td>
<td>1454.00</td>
<td>(1,1,1)</td>
</tr>
<tr>
<td>13</td>
<td>352x288</td>
<td>15.0000</td>
<td>1838.00</td>
<td>(1,2,0)</td>
</tr>
<tr>
<td>14</td>
<td>352x288</td>
<td>15.0000</td>
<td>1963.00</td>
<td>(1,2,1)</td>
</tr>
<tr>
<td>15</td>
<td>352x288</td>
<td>30.0000</td>
<td>2192.00</td>
<td>(1,3,0)</td>
</tr>
<tr>
<td>16</td>
<td>352x288</td>
<td>30.0000</td>
<td>2358.00</td>
<td>(1,3,1)</td>
</tr>
</tbody>
</table>
```

Each of the scalable layers indicated in the corresponding bit-stream extractor output (with Id’s equal to 0 to 16 in the Example 17) can be extracted using the option \(-sl\). Note that these scalable layers do not correspond to the usual notion of layer. For instance, in order to extract the scalable layer 7 in Example 17, which has a spatial resolution of 176x144 samples (QCIF), a frame rate of 15Hz, and a bit-rate of 631 kbit/s, the command in Example 18 can be used.

Example 18: Extraction of a scalable layer

```
BitStreamExtractorStatic input.svc output.svc –sl 7
```

The options \(-l\), \(-t\), and \(-f\) can be used to identify scalable layers by the DTQ values (see Example 17). The option \("-L\") specifies that all layers with dependency_id less than or equal to \(L\) shall be extracted, the option \("-T\") specifies that all layers with temporal_level less than or equal to \(T\) shall be extracted, and the option \("-F\") specifies that all layers with quality_level less than or equal to \(F\) shall be extracted. The Example 19 shows an alternative using the option \(-l\), \(-t\), and \(-f\) for extracting the scalable layer 7 (cp. Example 17 and Example 18).

Example 19: Extraction of a scalable sub-stream using the option \(-l\), \(-t\), and \(-f\)

```
BitStreamExtractorStatic input.svc output.svc –l 0 –t 2 –f 1
```

The use of \(-and\) with \(-f\) \(F\) and \(-l\) \(L\) options allows to extract all whole layers below layer \(L\) and quality level \(F\) of layer \(L\).

The option \("-b B\) can be used for extracting s sub-stream with a bit-rate specified by \(B\). The bit-rate \(B\) has to be specified in kbit/s. The extracted sub-stream is determined by the following ordered steps.

1. If there is a layer having a bitrate exactly equal to \(B\), that layer is extracted. Otherwise, the extraction process continues with step 2.
2. The layer \(X\) having a bitrate smaller than but closest to \(B\) is found. If layer \(X\) is the highest layer, layer \(X\) is extracted. Otherwise, the extraction process continues with step 3.
3. If the next higher layer \(Y\) to layer \(X\) is not an FGS layer, then layer \(X\) is extracted. Otherwise, the extraction process continues with step 4.
4. If the bitrate of layer \(Y\) is greater than \(B\), layer \(Y\) is truncated to \(B\). Otherwise, let layer \(X\) be replaced with layer \(Y\), and go to step 3.
The most powerful option for operating the bit-stream extractor is option “-e”. When using this option, the spatial resolution, the frame rate, and the bit-rate of the sub-stream to be extracted can be specified. This extraction option has to be specified in the form “-e AxB@C:D”. At this, the parameters A and B represent the spatial resolution expressed by the frame width A and the frame height B. The frame rate of the sub-stream to be extracted is specified by the parameter C, and the target bit-rate is specified by the parameter D. When FGS data are presents, the corresponding packets might be truncated in order to match the given target bit-rate. Note that each scalable stream only supports a specific range of target bit-rates for each spatio-temporal resolution, the supported bit-rate range is dependent on the encoder configuration. Example 20 shows an extractor call, in which a sub-stream in QCIF resolution (176x144 samples) with a frame rate of 15Hz and a bit-rate of 600 kbit/s is extracted. The extracted bit-stream is similar to that of Example 18 and Example 19 with the difference that the bit-rate (631 kbit/s in Example 18 and Example 19) is reduced to 600 kbit/s. The reduction of the bit-rate is obtained by truncating FGS packets (quality_level > 0) of the scalable layer 7 in Example 17.

Example 20: Extraction of a scalable sub-stream using the general option –e

```
BitStreamExtractorStatic input.svc output.svc -e 176x144@15:600
```

When using the general extraction option “-e”, an additional option “-ql” can be specified. When this option is specified the extractor uses the quality level information in the bit-stream to extract a rate-distortion optimized bit-stream. In general the quality (PSNR) of an extracted bit-stream is higher when the option “-ql” is used. However, this option should only be used when quality layer information are embedded in the bit-stream; otherwise, the results is undefined. More information on how quality level information can be embedded in a bit-stream are given in section 2.5.

Example 21: Extraction of a scalable sub-stream using -ql

```
BitStreamExtractorStatic input.svc output.svc -e 352x288@30:1400 -ql
```

If the quality level assigner did a multi layer quality layer assignment (see section 2.5), extraction using the option “-ql” may result in the removal of some of the lower layer’s(dependency_id’s) FGS packets before all the top layer’s FGS packets are removed. This behaviour is intentional, since the goal of multi layer quality layer assignment is to provide optimal RD performance at the top most layer (may be at the cost of the RD performance at embedded lower layers). However for certain applications, it may be good to remove lower layer’s(dependency_id’s) FGS packets only after all the top layer’s FGS packets have been removed. This ensures that the quality of the embedded spatial layers(dependecy_ids) are not affected by bit stream extraction, if the target bitrate allows it. This kind of ordered extraction can be performed by using the option “-qlord”.

Using “-qlord” has the same effect as using “-ql” if the quality layer assigner did not use multi layer quality layer assignment. Using “-qlord” has the same effect as using “-ql” if there is only one layer(dependency_id).

Example 22: Extraction of a scalable sub-stream using -qlord

```
BitStreamExtractorStatic input.svc output.svc -e 352x288@30:1400 -qlord
```

When using the general extraction option “-e”, an additional option “-sip” can be specified. When this option is specified the extractor uses the selective inter-layer prediction information in the bit-stream to extract a rate-distortion optimized bit-stream. This option should only be used when quality layer information are embedded in the bit-stream. If the base layer is AVC compatible, only when the base layer bit-stream is encoded with suffix units and the option “-suf” is present. And “-sip” can’t be used in connection with “-ql”. More information on how the selective inter-layer prediction information can be embedded in a bit-stream is given in section 2.9.

With the option “-pt” a so-called packet trace file can be generated from a given stream. This trace file is a text file, which specifies various parameters for each single “packet” inside the given bit-stream.
These parameters include the start position (in units of bytes) of the packet inside the bit-stream, the length of the packet (in units of bytes), the values of dependency_id (LId), temporal_level (TId), and quality_level (QId) for the packet, the type of the packet, and two flag which indicate whether the packet is discardable or truncatable. An example, which shows how such a trace file can be generated and what information is present in the trace file is given in Example 23.

The packet trace file can be modified and then used for bit-stream extraction. This is a very useful feature for easily simulating non-regular packet losses or packet truncations. In order to remove a packet from a trace file, simply delete the corresponding line in the text file. For simulating a packet truncation, just reduce the Length parameter in the trace file. Note that the parameter Start-Pos shall not be modified for a packet, since this parameter is used for identifying packets inside a bit-stream. Only the parameters Start-Pos and Length are used for a trace-file-based bit-stream extraction; all other parameters are irrelevant and can be arbitrarily modified or deleted. Example 24 shows a modified version of the trace file in Example 23. Here, several packets have been deleted and other packets have been truncated. The example further shows how this modified trace file can be used for extracting a bit-stream. The extracted bit-stream can for instance be used for testing the error robustness of the decoder.

**Example 23: Example for generating a packet trace file**

```
> BitStreamExtractorStatic -pt trace.txt input.svc
> type trace.txt
```

<table>
<thead>
<tr>
<th>Start-Pos.</th>
<th>Length</th>
<th>LId</th>
<th>TId</th>
<th>QId</th>
<th>Packet-Type</th>
<th>Discardable</th>
<th>Truncatable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>StreamHeader</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000000a2</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ParameterSet</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000000af</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ParameterSet</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000000b8</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ParameterSet</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000000c1</td>
<td>1408</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SliceData</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000000d1</td>
<td>1838</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x00000000df</td>
<td>2811</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000186</td>
<td>532</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SliceData</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x000000017e</td>
<td>1809</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000218</td>
<td>2837</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x00000002ca</td>
<td>232</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x000000028d</td>
<td>167</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000233</td>
<td>492</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000301</td>
<td>163</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x00000003c2</td>
<td>64</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000310</td>
<td>250</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x000000031f</td>
<td>181</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x00000003b1</td>
<td>77</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x000000032f</td>
<td>250</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x00000003f8</td>
<td>103</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x000000034f</td>
<td>27</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000347</td>
<td>138</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000350</td>
<td>115</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x0000000357</td>
<td>18</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000389</td>
<td>127</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000369</td>
<td>119</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x0000000367</td>
<td>19</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x00000003e2</td>
<td>154</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000372</td>
<td>123</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>0x000000037a</td>
<td>18</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x000000037b</td>
<td>122</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000383</td>
<td>1362</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SliceData</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0x00000003d8</td>
<td>1815</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x0000000449</td>
<td>2814</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>SliceData</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0x00000004f9</td>
<td>219</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>SliceData</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example 24: Example for extracting a bit-stream using a (modified) packet trace file**

```
> type trace_mod.txt
```

<table>
<thead>
<tr>
<th>Start-Pos.</th>
<th>Length</th>
<th>LId</th>
<th>TId</th>
<th>QId</th>
<th>Packet-Type</th>
<th>Discardable</th>
<th>Truncatable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>StreamHeader</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Example 25: Extraction of a ROI sub-stream using -r

BitStreamExtractorStatic input.svc output.svc -r 1 -e 352x288@30:1400

The option "-r" can be used with the general extraction option "-e" to extract ROI which is defined by FMO. In the example 25, "-r 1" means extract roi whose roi_id is 1.

2.5 Quality level assigner "QualityLevelAssignerStatic"

The quality level assigner can be used to embed information about quality layers inside a bit-stream. These quality layer information can then be employed by the extraction process as described in section 2.4 in order to optimize the rate-distortion efficiency of the extracted sub-stream. Example 26 illustrates the usage of the quality level assigner.

Example 26: Using the quality level assigner

QualityLevelAssignerStatic -in Input -org L Original [-org L Original] 

or

QualityLevelAssignerStatic -in Input -out Output -rp DatFile [-sei]

- in Input - input bit-stream
- out Output - output bit-stream with determined quality layer id's
- org L Original - original image sequence for layer L
- wp DatFile - data file for storing rate and distortion values
- rp DatFile - data file with previously computed rate and distortion values
- sei - provide quality layer info using SEI messages
- dep - determine only dependent distortions (speed-up by factor of 2, slight coding eff. losses)
- ind - determine only independent distortions (speed-up by factor of 2, slight coding eff. losses)
- mlql - determine Multi Layer Quality Layer Ids
The input bit-stream file is specified by the parameter Input. As input a bit-stream generated by the encoder shall be specified. The output bit-stream is given by the parameter Output. Input and output bit-stream are identical with the exception that the output bit-stream contains additional information about quality layers. The quality layer information can be present in two different ways: (1) These information can be embedded in the NAL unit header syntax element simple_priority_id. In this case, the bit-rate is kept constant. This method can only be applied when the 3-byte NAL unit header is used. (2) The quality layer information can be inserted as additional SEI messages. Due to the additional NAL units, the bit-rate of the output bit-stream is slightly increased in comparison to the input bit-stream, but this method can also be applied when the 2-byte NAL unit header is used. By default, the quality layer information are embedded in the NAL unit header syntax element simple_priority_id. When SEI messages should be inserted, the option “-sei” needs to be specified.

In any case an original video sequence has to be specified for each layer (spatial or CGS layer) that is present in the input bit-stream. The original video sequences are specified by the option “-org L org.yuv”, where L specifies the LayerId for a contained layer and “org.yuv” specifies the filename of the original video sequence. In Example 27, the usage of the quality level assigner is illustrated for a bit-stream that contains 2 spatial or CGS layers. The file “in.svc” represents the input bit-stream, and the files “LX.yuv” – with X being 0 or 1 – represent the original sequences for layer X. The output bit-stream is specified by the filename “out.svc”.

Example 27: Using the quality level assigner for inserting quality layer information into a 2-layer stream

```
QualityLevelAssigner -in in.svc -org 0 L0.yuv -org 1 L1.yuv -out out.svc
```

When the option “-sei” is additionally specified in the Example 27, the quality layer information would be inserted as SEI messages. Furthermore, it is possible to specify one of the options “-dep” and “-ind”. These option signal that a slightly modified algorithm should be used for determining the quality level information. By using one of these option, the execution time can be reduced by 50%, but the rate-distortion efficiency of quality layer information is usually slightly reduced.

Example 28: Using the quality level assigner in two steps and storing quality layer information to a file

```
QualityLevelAssigner -in in.svc -org 0 L0.yuv -org 1 L1.yuv -wp QLInfo.dat
QualityLevelAssigner -in in.svc -out out.svc -rp QLInfo.txt
```

The inserting of quality layer information can also be done in two steps as illustrated in Example 28. By specifying the option “-wp QLInfo.dat”, the quality layer information are stored into the file “QLInfo.dat”. And this file can later be used via the option “-rp QLInfo.dat” to directly insert the quality layer information into a bit-stream without carrying out the complex and time-consuming rate-distortion analysis. These options are for instance useful when two different bit-stream with quality layer information – one using the syntax element simple_priority_id and one using SEI messages – shall be generated as illustrated in Example 29.

Example 29: Using the quality level assigner for generating two streams with quality layer information

```
QualityLevelAssigner -in in.svc -org 0 L0.yuv -org 1 L1.yuv -wp QLInfo.dat
QualityLevelAssigner -in in.svc -out out_pid.svc -rp QLInfo.txt
QualityLevelAssigner -in in.svc -out out_sei.svc -rp QLInfo.txt -sei
```

By default, the Quality level assigner assigns Quality layer Ids in such a way that the Quality layer Ids associated with a lower layer(dependency_id) is in a higher range compared to the Quality layer Ids associated with a higher layer(dependency_id). This ensures that the extractor removes the FGS packets associated with the a higher layer(dependency_id) before it removes FGS packets of a lower layer(dependency_id).

Multi Layer Quality Layer assignement (as proposed in JVT-S043) can be done via the option “-mlql”. This may improve the top layer(dependency_id) PSNR when the extractor truncates the bitstream using the option "-ql".
This option is useful only if there is more than one layer(dependency id). Specifically, this option is targeted for the combined scalability (meaning, a configuration having more than one Spatial/CGS layer and their associated FGS layers) configuration. If this option is used, the Quality layer Id assignment is done for the best RD performance at the top layer(dependency_id). The Quality layer Ids associated with a particular layer(dependency_id) may take any valid value.

Example 30: Using the quality level assigner for generating multi layer quality layer information
```
QualityLevelAssigner -in in.svc -out out.svc -org 0 L0.yuv -org 1 L1.yuv -mlql
```

The extractor can choose to use the multi layer quality layer information directly or it can choose to use it for an ordered truncation. More information about this can be obtained by reading the extractor usage description above.

### 2.6 MCTF pre-processing tool “MCTFPreProcessorStatic”

The MCTF pre-processing tool can be used for pre-filtering image sequences. Example 31 illustrates the usage of the MCTF pre-processor.

Example 31: Using the MCTF pre-processor
```
MCTFPreProcessor -w Width -h Height -f frms -i Input -o Output
[-gop GOPSize] [-qp QP]

- w Width   - frame width in luma samples (multiple of 16)
- h Height   - frame height in luma samples (multiple of 16)
- f frms     - number of frames to be processed (>1)
- i Input    - input sequence
- i Output   - output sequence
- gop GOPSize - GOP size for MCTF (2,4,8,16,32,64, default: 16)
- qp QP      - QP for motion estimation and mode decision
  (>0, default: 26)
```

The input sequence is specified by the parameter Input. The output sequence is given by the parameter Output. The parameters width and height specify the frame width and heights in luma samples, respectively. The parameter frms specifies the number of frames that are pre-filtered. The GOP size, which is used for applying the MCTF pre-filter, is given by the parameter GOPSize. Allowed values for GOPSize are 2, 4, 8, 16, 32, and 64. The parameter QP specifies the Lagrangian multiplier, which is used for motion estimation and mode decision for the MCTF analysis. This parameter controls the “strength” of the pre-filter.

In Example 32, the usage of the MCTF pre-processor is illustrated for 300 frames of a sequence in 4CIF resolution. The file “in.svc” represents the input sequence, and the file “pre.yuv” specifies the output sequence. For the parameters GOPSize and QP the default values of 16 and 26 are used, respectively.

Example 32: Example for using the MCTF pre-processor
```
MCTFPreProcessor -in in.svc -out pre.yuv –w 352 –h 288 –f 300
```

### 2.7 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 33. The PSNR of every frame is written to stdout, and the average PSNR over all frames is written to stderr.

Example 33: 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 2.1.1.

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 bit-stream is additionally outputted when a bit-stream file `strm` and a frame rate (in Hz) `fps` is specified. It is not possible to specify only one of these parameters.

**Example 34: Using the PSNR tool for PSNR and rate measurement**
```
PSNRStatic 176 144 org.yuv rec.yuv 0 0 strm.svc 15 2>PSNR.txt
```
```
type PSNR.txt
128,00 32,23 38,79 39.02
```

In Example 34, 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 “strm.svc” specifies a bit-stream. 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.

### 2.8 Fixed QP encoder “FixedQPEncoderStatic”

The Fixed QP Encoder tool can be used to find the basis quantization parameter to meet a rate constraint for single layer coding, or in case of scalable coding, the rate constraints for the layers of a scalable stream. A logarithmic search algorithm is applied to iteratively determine the quantization parameter.

An example for the usage of the FixedQPEncoderStatic binary is depicted in Example 35 below.

**Example 35: Using the fixed qp encoder**
```
FixedQPEncoderStatic <rc_cfg>
```

The parameter `rc_cfg` represents the rate control parameter file configuring the fixed QP encoder. The configuration file reads specific parameters from defined lines of the configuration file. Therefore, the ordering of the parameters in the configuration file is fixed. For each layer, a target bit rate, the
maximum positive and negative bit-rate mismatch and a start value for the quantization parameter is specified. An example for the fixed qp encoder configuration file is provided in Example 36 below.

Example 36: Example fixed qp encoder configuration file for determination of the layer base quantization parameters with 2-layer encoding. For reference, the line numbers are printed before the lines.

```
1  ########### RATE-POINTS CONFIGURATION FILE ###########
2
3   CITY                              # Label
4   bin/H264AVCEncoderLibTestStatic   # Encoder Binary
5   cfg/CITY.cfg                      # Parameter File
6   str/CITY_P2.264                   # Bit Stream File
7   mot                               # Motion File Folder
8   600                               # Number Of Frames
9   60.0                              # Frames Per Second
10  2                                 # Number Of Layers
11  1                                 # constrained intra for base
12                                      layer (single-loop decoding)
13
14  ---------- LAYER 0 ----------
15  384.00                            # Bit rate [kbit/s]
16  2.00                              # Maximum Negative Mismatch [%]
17  2.00                              # Maximum Positive Mismatch [%]
18  29.00                             # Start QP
19
20  ---------- LAYER 1 ----------
21  1152.00                           # Bit rate [kbit/s]
22  2.00                              # Maximum Negative Mismatch [%]
23  2.00                              # Maximum Positive Mismatch [%]
24  33.00                             # Start QP
```

**Label (line 3)**

*String*

Label used for temporary file names (typically the name of the sequence under consideration)

**Encoder Binary (line 4)**

*String*

Name and path of the encoder binary to be used.

**Parameter File (line 5)**

*String*

Name of the main configuration file as specified in section 2.2 for the configuration to be optimized. The main configuration file specifies the maximum number of layers.

**Bit Stream File (line 6)**

*String*

Name of the results bit stream file.

**Motion File Folder (line 7)**

*String*

Name of the directory where the files with the motion data for each layer are stored.

**Number of Frames (line 8)**

*Unsigned Int*

Number of frames to be encoded at the target resolution. Overwrites the parameter FramesToBeEncoded in the main configuration file

**Frames Per Second (line 9)**

*Double*

Maximum frame rate of all input sequences or a multiple thereof. Overwrites the parameter FrameRate in the main configuration file.

**Number of Layers (line 10)**
Unsigned int
Specifies the number of layers for which parameters for the QP optimization are provided. The maximum number of layers is equal to the value of NumLayers in the main configuration file. A maximum number of 10 layers can be specified.

Constrained intra for base layer (line 11)
Unsigned int
Flag to indicate the usage of the command line parameter –bcip for the lowest layer to enable single loop decoding.

The description of the layer parameters is given for a layer nlayer, with nlayer being in the range of 0..10 (maximum).

Bit rate (line 15+6*nlayer)
Double
Target bit rate for layer nlayer in kbit/s.

Maximum Negative Mismatch (line 16+6*nlayer)
Double
Percentage of maximum rate under-shoot relative to the layer target bit rate.

Maximum Positive Mismatch (line 17+6*nlayer)
Double
Percentage of maximum rate over-shoot relative to the layer target bit rate.

Start QP (line 18+6*nlayer)
Double
Layer quantization parameter as specified in section 2.2.2.2 to start from. A good guess for this parameter may reduce the number of iterations to meet the target bit rate.

2.9 SIP Analyser tool “SIPAnalyser”

The SIPAnalyser is used to make the selective inter-layer prediction decision. The SIPAnalyser needs frames bits information got from encoder with option “-anasip” on, and outputs the SIP decision on each layer. A dynamic programming algorithm is used to make such a decision.

An example for the usage of the SIPAnalyser binary is depicted in Example 37 below.

Example 37: Using the SIP Analyser tool

SIPAnalyser.exe <sip_cfg> [FileLabel]

The parameter sip_cfg specifies the SIP parameter file. The configuration file reads specific parameters from defined lines of the configuration file. Therefore, the ordering of the parameters in the configuration file is fixed. For each layer, the out frame rate, the loss which can be tolerated under the condition with MA and the input and output filenames are specified. The parameter FileLabel add a suffix to the output filename. An example for the sip configuration file is provided in Example 38 below.

Example 38: Example main configuration file “main.cfg” for single-layer coding

| 1 | # Number of Layers |
| 2 | # Number of Frames |
| 3 | # In Frame Rate |
| 4 | # Out Frame Rate |
| 5 | # Dat File Without |
| 6 | # Dat File With |
| 7 | # Output SIP Decision (dummy for layer 0) |

15 1.03 # loss which can be tolerated with MA (dummy for layer 0)
16 30 # out frame rate
17 SIP/bus_1_without.dat # dat file without interlayer pred
18 SIP/bus_1_with.dat # dat file with interlayer pred (dummy for layer 0)
19 SIP/bus_1.dat # output SIP decision (dummy for layer 0)

Number of Layers (line 3)

Unsigned int
Specifies the number of layers for which parameters for the SIP decision are provided.

Number of Frames (line 4)

Unsigned Int
Specifies the number of frames to be encoded at the highest resolution.

In Frame Rate (line 5)

Unsigned Int
Specifies the input frame rate when encoding.

Toleration (line 8+7*nlayer)

Double
Specifies the loss which can be tolerated under the condition with MA (it’s dummy for layer 0).

Out Frame Rate (line 9+7*nlayer)

Unsigned Int
Specifies the output frame rate on this layer when encoding..

Dat File Without (line 10+6*nlayer)

String
Specifies the dat file without interlayer prediction (it’s dummy for layer 0).

Dat File With (line 11+6*nlayer)

String
Specifies the dat file with interlayer prediction (it’s dummy for layer 0).

Output SIP Decision (line 12+6*nlayer)

String
Specifies the output SIP decision file name (it’s dummy for layer 0).

3 Use Examples as a brief tutorial

In the following subsection, the usage of the JSVM 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 spatially and temporaly downscaled versions of the so-called original YUV sequences in 4CIF and CIF resolution. These original sequences may be downloaded via ftp.tnt.uni-hannover.de/pub/svc/testsequences/. Please note that whereas the above-mentioned FTP site already contains downscaled versions of the original YUV sequences at different spatial and temporal resolutions, these had been obtained using downsampling filters that do not correspond to the SVC normative filters. Using these sequences may result in incoherent and/or suboptimal results in term of coding-efficiency.

3.1.1 CIF scenario

3.1.1.1 Original sequences

The original CIF resolution sequences are the following:
• BUS_352x288_30_orig_01.yuv,
• FOREMAN_352x288_30_orig_01.yuv
• FOOTBALL_352x288_30_orig_01.yuv
• MOBILE_352x288_30.yuv.

For further ease of use, it is strongly suggested to rename these sequences as BUS_352x288_30.yuv, FOOTBALL_352x288_30.yuv, FOREMAN_352x288_30.yuv and MOBILE_352x288_30.yuv.

3.1.1.2 How to generate dowsampled versions

The so-called original YUV sequences in CIF resolutions have a type-1 chroma position (phase factor: 0, 0). Therefore, their downsampled versions may be obtained with the resampler tool (see section 2.1) using the following parameters:

Example 39: Down-sampling original CIF 30Hz sequence to QCIF 15Hz sequence for the CIF scenario.

```
DownConvertStatic 352 288 SEQ_352x288_30.yuv 176 144 SEQ_176x144_15.yuv 0 1 0 frms -phase 0 0
```

Where “SEQ” is the sequence name (i.e. “BUS”, “FOOTBALL”, “FOREMAN” or “MOBILE”) and “frms” is the number of frames wanted in output file.

It is worth noting that whereas the above-mentioned process corresponds to dyadic downsampling we recommend here to use method 0 (that corresponds to non-normative downsampling (JVT-R006)) instead of method 1.

3.1.1.3 How to consider the downsampling in the encoder configuration

If the downsampling method described above is applied in a scenario using spatial scalability, it has to be indicated in the configuration file for layer 1:

Example 40: ESS parameters for Layer1(CIF) configuration file with Extended Spatial Scalability.

```
UseESS          1         # ESS
ESSCropWidth    352       # cropping width
ESSCropHeight   288       # cropping height
ESSOriginX      0         # cropping origin X
ESSOriginY      0         # cropping origin Y
ESSChromaPhaseX 0         # chroma phase X 0 or -1, default = -1
ESSChromaPhaseY 0         # chroma phase Y -1 to +1, default = 0
ESSBaseChromaPhaseX 0     # base chroma phase X 0 or -1, default = -1
ESSBaseChromaPhaseY 0     # base chroma phase Y -1 to +1, default = 0 0
```

3.1.2 4CIF scenario

3.1.2.1 Original sequences

The original 4CIF resolution sequences are the following:

• CITY_704x576_60_orig_01.yuv
• CREW_704x576_60_orig_01.yuv
• HARBOUR_704x576_60_orig_01.yuv
• SOCCER_704x576_60_orig_02.yuv. (Note: zero two is correct!)

For further ease of use, it is strongly suggested to rename these sequences as CITY_704x576_60.yuv, CREW_704x576_60.yuv, HARBOUR_704x576_60.yuv, and SOCCER_704x576_60.yuv.
3.1.2.2 How to generate downsized samples

The so-called original YUV sequences in 4CIF resolutions have a type-0 chroma position (phase factor: -1, 0). Therefore, their downsized versions may be obtained with the resampler tool (see section 2.1) using the following parameters:

Example 41: Down-sampling original CIF 60Hz sequence to CIF 30Hz sequence for the 4CIF scenario.

```
DownConvertStatic 704 576 SEQ_704x576_60.yuv 352 288 SEQ_352x288_30.yuv 0 1 0 frms –phase -1 0
```

Example 42: Down-sampling CIF 30Hz sequence to QCIF 15Hz sequence for the 4CIF scenario.

```
DownConvertStatic 352 288 SEQ_352x288_30.yuv 176 144 SEQ_176x144_15.yuv 0 1 0 frms –phase -1 0
```

Where “SEQ” is the sequence name (i.e. “CITY”, “CREW”, “HARBOUR” or “SOCCER”) and “frms” is the number of frames wanted in output file.

It is worth noting that original sequences in QCIF resolutions are obtained via a two-steps process and not in a single one, which would result in different sequences. In addition, it is worth noting that whereas the above-mentioned process corresponds to dyadic downsampling, we recommend here to use method 0 (that corresponds to non-normative downsampling (JVT-R006)) instead of method 1.

3.1.2.3 How to consider the downsampling in the encoder configuration

If the downsampling method described above is applied in a scenario using spatial scalability, it has to be indicated in the configuration file for layer 1 and layer 2:

Example 43: ESS parameters for Layer1(CIF) configuration file with Extended Spatial Scalability.

```
UseESS          1         # ESS
ESSCropWidth    352       # cropping width
ESSCropHeight   288       # cropping height
ESSOriginX      0         # cropping origin X
ESSOriginY      0         # cropping origin Y
```

Example 44: ESS parameters for Layer1(CIF) configuration file with Extended Spatial Scalability.

```
UseESS          1         # ESS
ESSCropWidth    704       # cropping width
ESSCropHeight   576       # cropping height
ESSOriginX      0         # cropping origin X
ESSOriginY      0         # cropping origin Y
```

3.2 Single layer coding

As a first example, we look at single-layer coding with the JSVM software. For this example, the JSVM software is run in scalable mode for supporting temporal scalability. For more information on how the run the software in single-layer mode the reader is referred to section 2.2.1.

Single-layer coding means that the encoded bit-stream does not provide several spatial resolutions or several bit-rates for a specific spatio-temporal resolution. Thus, the generated bit-stream neither contains spatial or CGS enhancement layer nor additional progressive refinement slices (FGS slices).

Let’s assume, we want to encode a given sequence “BUS_QCIF15.yuv” in QCIF resolution (176x144 samples) with a frame rate of 15 Hz. Examples for the main and layer configuration files are depicted in Example 45 and Example 46, respectively.

Example 45: Example main configuration file “main.cfg” for single-layer coding

```
# JSVM Main Configuration File
OutputFile              test.264   # Bitstream file
FrameRate               30.0       # Maximum frame rate [Hz]
FramesToBeEncoded       150        # Number of frames (at input frame rate)
GOPSize                 16         # GOP Size (at maximum frame rate)
BaseLayerMode           2          # Base layer mode (0: AVC w larger DPB, # 1:AVC compatible, 2:AVC subseq SEI)
```
The most important parameters that need to be specified in the main configuration file are the name for the bit-stream *OutputFile*, the frame rate *FrameRate*, the number of frames to be encoded *FramesToBeEncoded*, the GOP size *GOPSize*, and the base layer mode *BaseLayerMode*. Furthermore, the parameter *NumLayers* has to be set equal to 1 for single-layer coding, and exactly one layer configuration file has to be specified via the parameter *LayerCfg*. In Example 45, we additionally specified *SearchMode* and *SearchRange* to speed-up the encoder execution. For all other parameters of the main configuration file, default values as specified in section 2.2.2.1 are used.

In the layer configuration file, the filename of the input sequence *InputFile*, the frame width *SourceWidth* and the frame height *SourceHeight* of the input images as well as the frame rates *FrameRateIn* and *FrameRateOut* need to be specified.

**Example 46: Example layer configuration file “layer.cfg” for single-layer coding**

```plaintext
# JSVM Layer Configuration File
InputFile            BUS_QCIF15.yuv # Input file
SourceWidth          176            # Input frame width
SourceHeight         144            # Input frame height
FrameRateIn          15             # Input frame rate [Hz]
FrameRateOut         15             # Output frame rate [Hz]
```

Note that the frame rates *FrameRate* and *FrameRateOut* are different. Thus, only 75 frames are actually encoded although the parameter *FramesToBeEncoded* is set equal to 150. Similarly, groups of 8 pictures are actually used for encoding the sequence “BUS_QCIF15.yuv”, although a GOP size *GOPSize* of 16 is specified in the main configuration file.

Note further that the base layer mode *BaseLayerMode* is set equal to 2. Hence, an AVC compatible bit-stream with additional sub-sequence SEI messages is written. These SEI messages are required for enabling the extraction of temporally downsampled sub-streams by using the bit-stream extractor. The employed coding structure does always provide temporal scalability, but information about what picture belongs to what temporal level are only present either when sub-sequence SEI messages are inserted into an AVC compatible stream or when the SVC mode (SVC NAL units) is used.

As mentioned above, even in single-layer mode, the generated bit-stream can provide temporal scalability. The number of supported temporal scalability levels is dependent on the specified GOP size. A so-called group of pictures (GOP) consists of a key picture and hierarchically predicted B pictures that are located between the key picture of the current GOP and the key picture of the previous GOP. In Figure 1, the hierarchical prediction structure is illustrated for a group of 8 pictures, which is employed in the example above.
Figure 1: Hierarchical prediction structure for groups of 8 pictures.

The key pictures are either intra-coded (e.g. in order to enable random access) or inter-coded by using previous key pictures as reference for motion-compensated prediction. The sequence of key pictures is independent from any other pictures of the video sequence, and in general it represents the minimal temporal resolution that can be decoded. Furthermore, the key pictures can be considered as re-synchronisation points between encoder and decoder when FGS enhancement layers are added. 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 45 and Example 46, the encoder can be started using the call in Example 47. With option “-pf” the main configuration file is specified. The option “-lqp 0 30” specifies the basis quantization parameter as well as the Lagrangian parameters for motion estimation and mode decision as described in section 2.2.2. Note that for rate-distortion efficient coding, the quantization parameter and the QP values that are used for determining the Lagrangian parameters should be identical. This is ensured by using the command line option “-lqp”.

Example 47: Example encoder call for single-layer coding

```
> H264AVCEncoderLibTestStatic -pf main.cfg -lqp 0 30
...
```

<table>
<thead>
<tr>
<th>SUMMARY:</th>
<th>SNR Level bitrate</th>
<th>Y-PSNR</th>
<th>U-PSNR</th>
<th>V-PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>176x144 @ 1.8750</td>
<td>0.0000</td>
<td>70.7250</td>
<td>37.0235</td>
<td>41.2961</td>
</tr>
<tr>
<td>176x144 @ 3.7500</td>
<td>0.0000</td>
<td>104.2689</td>
<td>36.2552</td>
<td>41.1378</td>
</tr>
<tr>
<td>176x144 @ 7.5000</td>
<td>0.0000</td>
<td>142.6216</td>
<td>35.5460</td>
<td>40.9821</td>
</tr>
<tr>
<td>176x144 @ 15.0000</td>
<td>0.0000</td>
<td>186.3536</td>
<td>34.8854</td>
<td>40.9277</td>
</tr>
</tbody>
</table>

At the end of Example 47, the final encoder output is shown. It summarizes the supported spatial resolutions, frame rates, and bit-rates. For our example, only a single spatial resolution of 176x144 samples is supported. But the bit-stream provides 4 different temporal resolutions with frame rates of 1.87, 3.75, 7.5, and 15Hz, since we are using an effective GOP size of 8 pictures. The lowest supported temporal resolution is build by the sequence of key pictures.

Example 48: Example of supported scalable layers for single-layer coding

```
> BitStreamExtractorStatic test.264
```

Contained Layers:
A printout of the supported spatio-temporal resolutions and bit-rates can also be obtained when the bit-stream extractor is called with the filename of the bit-stream as illustrated in Example 48. A sub-bitstream that only contains NAL units for a specific temporal resolution can be extracted from the global bit-stream “test.264” by using the bit-stream extractor either with option “-sl” or “-t” (cp. section 2.4. During the extraction of a sub-stream, all NAL units (packets) that are not required for decoding a specific spatio-temporal-rate point are simply discarded.

In Example 49, it is illustrated how the temporal sub-stream of 3.75 Hz is extracted from the global 30 Hz stream and how this sub-stream is decoded. When referring to Figure 1, this temporal sub-stream only consists of the (brown) key pictures and the (blue) pictures of the next temporal level. Furthermore in Example 49, the PSNR of the decoded video sequence and the bit-rate of the extracted sub-stream are measured using the PSNR tool, which is described in section 2.6. It shall be noted that the measured values for the bit-rate and the PSNR’s are identical to those that have been reported in the encoder output (see Example 47). The extremely minor differences in the rate values results from the fact, that the bits that are required for transmitting the stream header (scalable SEI message) are not counted during encoding.

Example 49: Example for extracting and decoding a temporal sub-sequence

```plaintext
>BitStreamExtractorStatic    test.264 substream.264 –t 1
>H264AVCDecoderLibTestStatic substream.264 dec.yuv
>PSNRStatic 176 144 BUS_QCIF15.yuv dec.yuv 0 2 substream.264 15 2>PSNR.dat
>type PSNR.dat
104,3211        36,2552 41,1378 42,4513
```

### 3.3 Spatial scalability

As next example we consider simple spatial scalable coding with two spatial resolutions, QCIF and CIF. Examples for the main configuration files and the two layer configuration files are depicted in Example 50, Example 51, and Example 52. The main configuration file is nearly identical to the one that was used for single-layer coding (cp. Example 45), and the configuration file for layer 0 is identical to the one in Example 46. Thus, the base layer (layer 0) represents a QCIF sequence with a frame rate of 15 Hz. In the enhancement layer, a CIF sequence with a frame rate of 30 Hz is coded. Consequently, the parameters `SourceWidth`, `SourceHeight`, `FrameRateIn`, and `FrameRateOut` are correspondingly modified for the configuration file for layer 1.

Additionally, the parameter `InterLayerPred` is specified in the configuration for the spatial enhancement layer. `InterLayerPred` equal to 2 specifies that the base layer (layer 0) is employed for inter-layer prediction, and that the actually used inter-layer prediction concepts are adaptively switch on a macroblock basis via rate-distortion-optimized coding. For rate-distortion efficient coding, `InterLayerPred` should be set equal to 2 for any enhancement layer.

Example 50: Example main configuration file “main.cfg” for spatial scalable coding

```plaintext
# JSVM Main Configuration File
```
OutputFile              test.264   # Bitstream file
FrameRate               30.0       # Maximum frame rate [Hz]
FramesToBeEncoded       150        # Number of frames (at input frame rate)
GOPSize                 16         # GOP Size (at maximum frame rate)
BaseLayerMode           2          # Base layer mode (0: AVC w larger DPB,
                              # 1:AVC compatible, 2:AVC w subseq SEI)
SearchMode              4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchRange             32         # Search range (Full Pel)
NumLayers               2          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
LayerCfg                layer1.cfg # Layer configuration file

Example 51: Example layer configuration file “layer0.cfg” for spatial scalable coding

# JSVM Layer Configuration File
InputFile            BUS_QCIF15.yuv # Input  file
SourceWidth          176            # Input  frame width
SourceHeight         144            # Input  frame height
FrameRateIn          15             # Input  frame rate [Hz]
FrameRateOut         15             # Output frame rate [Hz]

Example 52: Example layer configuration file “layer1.cfg” for spatial scalable coding

# JSVM Layer Configuration File
InputFile            BUS_CIF30.yuv  # Input  file
SourceWidth          352            # Input  frame width
SourceHeight         288            # Input  frame height
FrameRateIn          30             # Input  frame rate [Hz]
FrameRateOut         30             # Output frame rate [Hz]
InterLayerPred       2              # Inter-layer Pred. (0: no, 1: yes, 2:adapt.)

The hierarchical coding structure with 2 spatial layers is illustrated in Figure 2. In each layer (QCIF and CIF in the above example), an independent hierarchical coding structure with layer specific motion parameters is employed. Note that in the above example, the QCIF layer is coded at a frame rate of 15 Hz, while the CIF layer is coded at a frame rate of 30 Hz. In order to obtain a temporal scalable representation, the prediction structures of all layers have to be aligned as depicted in Figure 2. And when specifying a GOP size of 16 pictures in the main configuration file, the effective GOP sizes that are used for the QCIF and CIF layer are 8 and 16, respectively (in Figure 2, the prediction structure is illustrated for GOP sizes of 4 and 8). The red arrows in Figure 2 indicated the usage of inter-layer prediction. Inter-layer prediction can only be used inside an access unit, and thus between base and enhancement layer pictures at the same time instant. Since, the frame rate of the CIF enhancement layer is twice the frame rate of the QCIF base layer, the enhancement layer pictures of the highest temporal level are coded without inter-layer prediction. These pictures are only predicted using motion-compensated temporal prediction.
Given the configuration files of Example 50, Example 51, and Example 52, an example encoder call is illustrated in Example 53. It is assumed that the original sequence “BUS_CIF30.yuv” is given in CIF resolution with a frame rate of 30 Hz. In a first step, the resampling tool is used for generating a spatially and temporally downsampled sequence “BUS_QCIF15.yuv” in QCIF resolution with a frame rate of 15 Hz. More information about the resampling tool are given in section 2.1. Then the encoder is called with the main configuration file “main.cfg” and the additional options “-lqp 0 30” and “-lqp 1 32”, where the first values (0 and 1) specify the layer, and the second values (30 and 32) specify the corresponding quantization parameter. As described in section 3.1, with option “-lqp” the quantization parameter as well as Langrangian parameters for motion estimation and mode decision are specified. The bit-rate of the generated sequence is mainly dependent on these parameters. Similar to single-layer coding the quantization parameter $Q_P$ and the parameters $MeQPX$, which determine the Lagrangian multipliers for mode decision and motion estimation, should be set to identical values for each spatial layer, in order to generated a rate-distortion optimized bit-stream. This is always ensured by using the command line option “-lqp”.

**Example 53: Example encoder call for spatial scalable coding**

```
>DownConvertStatic 352 288 BUS_CIF30.yuv 176 144 BUS_QCIF15.yuv 0 1

>h264avcencoderlibteststatic -pf main.cfg -lqp 0 30 -lqp 1 32

... SUMMARY:

<table>
<thead>
<tr>
<th>SNR Level bitrate</th>
<th>Y-PSNR</th>
<th>U-PSNR</th>
<th>V-PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>176x144 @ 1.8750</td>
<td>71.0145</td>
<td>36.9930</td>
<td>41.3140</td>
</tr>
<tr>
<td>176x144 @ 3.7500</td>
<td>104.8926</td>
<td>36.2550</td>
<td>41.1325</td>
</tr>
<tr>
<td>176x144 @ 7.5000</td>
<td>143.5089</td>
<td>35.5507</td>
<td>40.9885</td>
</tr>
<tr>
<td>176x144 @ 15.0000</td>
<td>187.6496</td>
<td>34.8944</td>
<td>40.9219</td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>299.4675</td>
<td>37.3048</td>
<td>41.7122</td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>412.7953</td>
<td>35.9055</td>
<td>41.3231</td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>536.1900</td>
<td>34.8822</td>
<td>41.0904</td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>682.5856</td>
<td>34.1586</td>
<td>40.9484</td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>809.1168</td>
<td>33.5824</td>
<td>40.7962</td>
</tr>
</tbody>
</table>
```

At the end of example Example 53, the corresponding encoder output is shown, which summarizes the supported spatio-temporal resolutions and bit-rates. This information can also be obtained by calling
the bit-stream extractor with the generated bit-stream as shown in Example 54. It should be noted that the lowest supported frame rate of 1.875 Hz is identical to the frequency of key pictures.

Example 54: Example of supported scalable layers for spatial scalable coding

```plaintext
>BitStreamExtractorStatic test.264

Member Layers:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Resolution</th>
<th>Framerate</th>
<th>Bitrate</th>
<th>DTQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>176x144</td>
<td>1.8750</td>
<td>71.00</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>176x144</td>
<td>3.7500</td>
<td>105.00</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>2</td>
<td>176x144</td>
<td>7.5000</td>
<td>144.00</td>
<td>(0,2,0)</td>
</tr>
<tr>
<td>3</td>
<td>176x144</td>
<td>15.0000</td>
<td>188.00</td>
<td>(0,3,0)</td>
</tr>
<tr>
<td>4</td>
<td>352x288</td>
<td>1.8750</td>
<td>299.00</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>5</td>
<td>352x288</td>
<td>3.7500</td>
<td>413.00</td>
<td>(1,1,0)</td>
</tr>
<tr>
<td>6</td>
<td>352x288</td>
<td>7.5000</td>
<td>536.00</td>
<td>(1,2,0)</td>
</tr>
<tr>
<td>7</td>
<td>352x288</td>
<td>15.0000</td>
<td>683.00</td>
<td>(1,3,0)</td>
</tr>
<tr>
<td>8</td>
<td>352x288</td>
<td>30.0000</td>
<td>809.00</td>
<td>(1,4,0)</td>
</tr>
</tbody>
</table>
```

The generated scalable bit-stream contains 9 different representations, 4 of them are QCIF representation with frame rates of 1.875, 3.75, 7.5, and 15 Hz, and 5 of them are CIF representations with frame rates of 1.875, 3.75, 7.5, 15, and 30 Hz. All of the included QCIF representations are AVC compatible and can be decoded with a standard AVC decoder. Sub-streams for any of the 9 included representations can be extracted using the bit-stream extractor. The bit-stream extractor can be called either with the option “-sl” or with the options “-l” and “-t”. In Example 55, examples for extracting a QCIF representation with a frame rate of 7.5 Hz (scalable layer 2 in the encoder output) and a CIF representation with a frame rate of 15 Hz are given. Note, that an extracted representation still includes all representations with a spatial resolution equal to or smaller to the spatial resolution of the extracted representation and a frame rate equal to or smaller than the frame rate of the extracted representation. Thus, the extracted representation can also be used for further extractions. For instance, the bit-stream “sub.264” obtained with

```plaintext
>BitStreamExtractorStatic subP1.264 sub.264 -1 0 -t 2
```

is identical to the bit-stream “subP0.yuv” of Example 55. After the extraction of spatio-temporal sub-streams in Example 55, these sub-streams are decoded, and the PSNR and the rate are measured similar to Example 49 for single-layer coding. Note again that the measured rate and PSNR values are identical to the values that are reported in the encoder output, with the exception of minor rate differences. The reason for obtaining these rate differences has been described in section 3.1.

Example 55: Example for extracting and decoding a spatio-temporal sub-sequence

```plaintext
>BitStreamExtractorStatic test.264 subP0.264 -l 0 -t 2
>BitStreamExtractorStatic test.264 subP1.264 -l 1 -t 3
>h264AVCDecoderLibTestStatic subP0.264 decP0.yuv
>h264AVCDecoderLibTestStatic subP1.264 decP1.yuv
>PSNRStatic 352 288 BUS_CIF30.yuv decP0.yuv 1 2 subP0.264 30 2> PSNR.dat
>PSNRStatic 352 288 BUS_CIF30.yuv decP1.yuv 0 1 subP1.264 30 2>>PSNR.dat
>type PSNR.dat
143,5816 35,5507 40,9885 42,3923
682,8032 34,1586 40,9484 42,9109
```
3.4 SNR scalability

In SVC, two different possibilities exist for providing SNR scalable bit-streams. The first one is referred to as coarse-grain scalability (CGS) and the second one is referred to as fine-grain scalability (FGS). In principle, CGS is identical to spatial scalable coding with the only exception that all layers have an identical spatial resolution. Thus, for generating a CGS bit-stream the examples in section 3.3 can be used, only the spatial resolution (SourceWidth and SourceHeight) of the layers has to be modified in a way that SourceWidth and SourceHeight are identical for all layers. Similar to spatial scalable coding, only a limited set of extractable rate points is included in a bit-stream.

In contrast to CGS coding, FGS coding supports a rate interval instead of a limited set of rate points, and a sub-stream for each rate inside the supported interval can be extracted. FGS coding is realized via so-called progressive refinement (PR) slices. The coding symbols inside the PR slices are ordered by their importance, and thus PR slices can be truncated at any point.

In Example 56 and Example 57, example configuration files for SNR scalable coding with PR slices are depicted. The main difference to the configuration files for single-layer coding (see section 3.1) is that the layer configuration file contains an additional parameter NumFGSLayers, which is set equal to 1. This parameter specifies that for each picture, one additional PR slice is transmitted. The PR slices refine the transform coefficients that are transmitted in the base layer, and thus increase the reconstruction quality of the pictures. A complete (not truncated) PR slice corresponds to a quality improvement that would be obtained when decreasing the quantization parameter by a value of 6. Additionally, the parameter ClosedLoop is set equal to 2 in the layer configuration file, which specifies that the encoder control is operated by closing the prediction loop at the lowest and the highest rate point.

Example 56: Example main configuration file “main.cfg” for SNR scalable coding

```plaintext
# JSVM Main Configuration File
OutputFile              test.264   # Bitstream file
FrameRate               30.0       # Maximum frame rate [Hz]
FramesToBeEncoded       150        # Number of frames (at input frame rate)
GOPSizze                16         # GOP Size (at maximum frame rate)
BaseLayerMode           2          # Base layer mode (0: AVC w larger DPB, 1:AVC compatible, 2:AVC w subseq SEI)
SearchMode              4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchRange             32         # Search range (Full Pel)
NumLayers               1          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
```

Example 57: Example layer configuration file “layer.cfg” for SNR scalable coding

```plaintext
# JSVM Layer Configuration File
InputFile              BUS_CIF30.yuv  # Input file
SourceWidth            352         # Input frame width
SourceHeight           288         # Input frame height
FrameRateIn            30          # Input frame rate [Hz]
FrameRateOut           30          # Output frame rate [Hz]
ClosedLoop             2           # Closed-loop control (0,1:at highest rate, 2: at lowest and highest rate)
NumFGSLayers           1           # Number of FGS layers ( 1 layer -> DQP = 6 )
```

Example 58 illustrates an possible encoder call for generating a SNR scalable bit-stream with the given configuration files. In addition to the option “-pf” (main configuration file), two options “-lqp” and “-rqp” are specified. The first option “-lqp” is known from the examples for single-layer and spatial scalable coding (section 3.1 and 3.3). This option specifies the quantization parameter and the Lagrangian parameters for mode decision and motion estimation. The second option “-rqp” specifies the quantization parameter. Thus, the previous value given by “-lqp” is overwritten. As a consequence, with option “-lqp” only the Lagrangian multipliers for motion estimation and mode decision are specified (as a QP equivalent), while the actual used quantization parameter is specified by the option
“-rqp”. For rate-distortion efficient FGS coding, the values of MeQP (option “-lqp” in the example) that are used for determining the Lagrangian parameters should be set to smaller values than the actual quantization parameter (option “-rqp” in the example). The “optimal” difference depends on the number of FGS layers that should be encoded as well as on the sequence content. For SNR scalable coding with one FGS layer, a QP difference of 1 or 2 is reasonable in most cases.

**Example 58: Example encoder call for SNR scalable coding**

```bash
>H264AVCEncoderLibTestStatic -pf main.cfg -lqp 0 30 -rqp 0 32
```

**SUMMARY:**

<table>
<thead>
<tr>
<th>SNR Level</th>
<th>bitrate</th>
<th>Y-PSNR</th>
<th>U-PSNR</th>
<th>V-PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>352x288 @ 1.8750</td>
<td>0.0000</td>
<td>269.00</td>
<td>42.1962</td>
<td>45.1907</td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>1.0000</td>
<td>465.9735</td>
<td>40.3225</td>
<td>44.3302</td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>0.0000</td>
<td>367.5079</td>
<td>46.0532</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>1.0000</td>
<td>658.7716</td>
<td>40.3225</td>
<td>44.3302</td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>0.0000</td>
<td>466.0532</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>1.0000</td>
<td>877.5411</td>
<td>38.9244</td>
<td>43.7575</td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>0.0000</td>
<td>586.2400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>1.0000</td>
<td>1159.4224</td>
<td>37.8919</td>
<td>43.3025</td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>0.0000</td>
<td>721.0688</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>1.0000</td>
<td>1474.8384</td>
<td>37.0108</td>
<td>42.8937</td>
</tr>
</tbody>
</table>

**Example 59: Example of supported scalable layers for SNR scalable coding**

```bash
>BitStreamExtractorStatic test.264
```

**Contained Layers:**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Resolution</th>
<th>Framerate</th>
<th>Bitrate</th>
<th>DTQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>352x288</td>
<td>1.8750</td>
<td>269.00</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>352x288</td>
<td>1.8750</td>
<td>466.00</td>
<td>(0,0,1)</td>
</tr>
<tr>
<td>2</td>
<td>352x288</td>
<td>3.7500</td>
<td>368.00</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>3</td>
<td>352x288</td>
<td>3.7500</td>
<td>659.00</td>
<td>(0,1,1)</td>
</tr>
<tr>
<td>4</td>
<td>352x288</td>
<td>7.5000</td>
<td>466.00</td>
<td>(0,2,0)</td>
</tr>
<tr>
<td>5</td>
<td>352x288</td>
<td>7.5000</td>
<td>878.00</td>
<td>(0,2,1)</td>
</tr>
<tr>
<td>6</td>
<td>352x288</td>
<td>15.0000</td>
<td>586.00</td>
<td>(0,3,0)</td>
</tr>
<tr>
<td>7</td>
<td>352x288</td>
<td>15.0000</td>
<td>1159.00</td>
<td>(0,3,1)</td>
</tr>
<tr>
<td>8</td>
<td>352x288</td>
<td>30.0000</td>
<td>721.00</td>
<td>(0,4,0)</td>
</tr>
<tr>
<td>9</td>
<td>352x288</td>
<td>30.0000</td>
<td>1475.00</td>
<td>(0,4,1)</td>
</tr>
</tbody>
</table>

At the end of Example 58 the encoder output is depicted, which corresponds to the output of the bit-stream extractor in Example 59. Although in these printouts only 2 layers are shown for each temporal resolution, these values should be interpreted as the minimum and maximum rate of the supported rate interval. When we look at a frame rate of 30 Hz, all bit-rates inside the interval from 721 kbit/s to 1475 kbit/s can be extracted from the generated bit-stream. The first bit-rate (SNR level 0) represents the bit-rate of the base layer and is identical to the minimum extractable bit-rate. The second rate (SNR level 1) represents the bit-rate of the base-layer plus that of all PR slices of the first FGS layer.

In Example 60, it is illustrated that it is possible to extract any rate inside this interval. In this example, a target rate of 768 kbit/s is specified via the command line option “-e” for the bit-stream extractor. And the rate that is measured by the PSNR tools verifies that this bit-rate was really extracted.

**Example 60: Example for extracting and decoding a sub-sequence in SNR scalable coding**

```bash
>BitStreamExtractorStatic test.264 substream.264 -e 352x288@15:768

>H264AVCDecoderLibTestStatic substream.264 dec.yuv

>PSNRStatic 352 288 BUS_CIF30.yuv dec.yuv 0 1 substream.264 30 2>PSNR.dat

>type PSNR.dat
```
With the above example the supported bit-rate range could only be adjusted by choosing a specific quantization parameter for the base layer and a number of FGS enhancement layers. But with the JSVM software it is also possible to perform an encoder-side truncation of PR slices, so that the maximum supported bit-rate can be specified by command line options.

Lets assume we want to support a bit-rate range from 700 to 2000 kbit/s. The lowest achievable rate can only be controlled via quantization parameters. Since, the lowest rate in the above example was 721 kbit/s, we have to increase the quantization parameter. This is done by using the command line option “-rqp 0 33” instead of “-rqp 0 32” as illustrated in Example 61. The encoding is now performed in two steps. For the first encoder run, the option “-anafgs 0 2 fgs.dat” specifies that 2 FGS layers are encoded and that information about the FGS packets are stored into the file “fgs.dat”. With this encoder run we only determine information about FGS packets that are employed in a second run for meeting a given target rate. For the second run, the option “-encfgs 0 2000 fgs.dat” specifies that now the FGS packets are truncated at the encoder side, so that a maximum bit-rate of 2000 kbit/s is achieved. The data in the file “fgs.dat” that have been generated during the first encoder run are used for this purpose. The encoder output at the end of Example 61 shows that indeed the maximum supported bit-rate is 2000 kbit/s. Note that in this case, two FGS layers have been encoded, but the second FGS layer has been additionally truncated during encoding. The supported bit-rate range for a temporal resolution of 30 Hz reaches from 654 to 2000 kbit/s and matches the above defined target range. For more information about the encoding options “-anafgs” and “-encfgs”, the reader is referred to section 2.2.2.3.

**Example 61: Example for SNR scalable encoding with a fixed maximum bit-rate**

```
> H264AVCEncoderLibTestStatic -pf main.cfg -lqp 0 30 -rqp 0 33 \
    -anafgs 0 2 fgs.dat
> H264AVCEncoderLibTestStatic -pf main.cfg -lqp 0 30 -rqp 0 33 \ 
    -encfgs 0 2000 fgs.dat
...
SUMMARY:
<table>
<thead>
<tr>
<th>SNR Level bitrate</th>
<th>Y-PSNR</th>
<th>U-PSNR</th>
<th>V-PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>352x288 @ 1.8750</td>
<td>0.0000</td>
<td>250.1250</td>
<td>250.1250</td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>1.0000</td>
<td>438.5070</td>
<td>438.5070</td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>2.0000</td>
<td>562.3470</td>
<td>435.3524</td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>0.0000</td>
<td>338.7584</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>1.0000</td>
<td>607.8016</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>2.0000</td>
<td>812.0116</td>
<td>41.5553</td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>0.0000</td>
<td>426.8621</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>1.0000</td>
<td>794.8437</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>2.0000</td>
<td>1115.6479</td>
<td>40.0338</td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>0.0000</td>
<td>533.4496</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>1.0000</td>
<td>1029.6928</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>2.0000</td>
<td>1521.9472</td>
<td>38.8880</td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>0.0000</td>
<td>653.5504</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>1.0000</td>
<td>1285.5392</td>
<td></td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>2.0000</td>
<td>1999.9984</td>
<td>37.9120</td>
</tr>
</tbody>
</table>
```

The rate-distortion efficiency of sub-streams that are extracted from an SNR scalable bit-stream can be increased when quality layer information are inserted into the bit-stream. Example 62 shows how the quality level assigner can be used for this purpose. The input bit-stream “test.264”, which has been generated by the encoder is investigated by a rate-distortion analysis, for which the original sequence “BUS_CIF30.yuv” is required. The determined quality layer information, which can be employed by the bit-stream extractor, are inserted into the newly created output bit-stream “testQL.264”. Beside the additional quality layer information, both bit-streams “test.264” and “testQL.264” are identical. More information about the quality level assigner are given in section 2.5.
Example 62: Example for using the quality level assigner for optimizing an SNR scalable bit-stream

```
>QualityLevelAssignerStatic -in test.264 -org 0 BUS_CIF30.yuv -out testQL.264
```

In the following, it is shown that the additional quality information really improve the rate-distortion efficiency of the extracted sub-streams. Example 63 and Example 64 show two batch files for extracting and decoding a set of 15 rate points from a SNR scalable bit-stream. And in Example 65, these batch scripts are used for generating sets of R-D values. For the first set of R-D values “Standard.dat”, the standard method for extracting rate points is employed with the original bit-stream “test.264”. An identical set of R-D values is obtained when the bit-stream “test.264” is replaced by the bit-stream “testQL.264”. For the second set of R-D values “QualityLayer.dat”, the quality layer information that have been inserted in the bit-stream “testQL.264” is employed for an optimized extraction of rate points. The optimized extraction method is specified by the option “-ql”. Note, however that it is not possible to use this option with a bit-stream that does not contain information about quality layers.

Example 63: Batch file decodePoint.bat for extracting and decoding a rate point of a SNR scalable stream.

```
:: (1:stream) (2:rate) (3:-ql)
BitStreamExtractorStatic %1 tmp.264 -e 352x288@30:%2 %3
H264AVCDecoderLibTestStatic tmp.264 tmp.yuv
PSNRStatic 352 288 BUS_CIF30.yuv tmp.yuv 0 0 0 tmp.264 30
```

Example 64: Batch file decode.bat for extracting and decoding a set of rate points of a SNR scalable stream.

```
:: (1:stream) (2:-ql)
FOR /L %%i IN (600,100,2000) DO @CALL decodePoint.bat %1 %%i %2
```

Example 65: Example for generating rate-distortion curves for SNR scalable bit-streams.

```
>decode.bat test.264 2>Standard.dat
>decode.bat testQL.264 -ql 2>QualityLayer.dat
```

In Figure 2, the generated sets of R-D points are plotted inside a diagram. It can be clearly seen, that the rate-distortion efficiency is improved when using the quality layer information. For the lowest and highest rate point, the coding efficiency is identical.
**3.5 Combined scalability**

The concepts for spatial and SNR scalable coding that have been used in section 3.3 and 3.4 can be combined for generating a bit-stream that supports a variety of spatio-temporal resolutions and rate points. In the next example we consider combined scalable coding with 3 spatial resolutions: QCIF, CIF, and 4CIF. The corresponding configuration files are given in Example 66, Example 67, Example 68, and Example 69. The structure of the configuration files is identical to those that have been used in section 3.3 and 3.4. One minor difference is that this time, the quantization parameter and the Lagrangian parameters for motion estimation and mode decision are directly specified in the layer configuration files via the parameters $Q_P$ and $MeQPX$ (with $X$ being equal to 0, 1, 2, 3, 4, and 5). Note that for the layers 0 and 1, no FGS layers have been specified. The FGS layers for QCIF and CIF will be specified by the command line options “-anafgs” and “-encfgs”. Note further that this time all spatial layers are encoded at an identical frame rate of 30 Hz.

**Example 66: Example main configuration file “main.cfg” for combined scalable coding**

```
# JSVM Main Configuration File
OutputFile              test.264   # Bitstream file
FrameRate               30.0       # Maximum frame rate [Hz]
FramesToBeEncoded       150        # Number of frames (at input frame rate)
GOPSize                 16         # GOP Size (at maximum frame rate)
BaseLayerMode           0          # Base layer mode (0: AVC w larger DPB, 
                                  # 1:AVC compatible, 2:AVC w subseq SEI)
SearchMode              4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchRange             32         # Search range (Full Pel)
NumLayers               3          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
LayerCfg                layer1.cfg # Layer configuration file
```
Example 67: Example layer configuration file “layer0.cfg” for combined scalable coding

```
# JSVM Layer Configuration File

InputFile        SOCCER_QCIF30.yuv # Input file
SourceWidth         176            # Input frame width
SourceHeight        144            # Input frame height
FrameRateIn         30             # Input frame rate [Hz]
FrameRateOut        30             # Output frame rate [Hz]
ClosedLoop          2              # Closed-loop control (0,1:at highest
#     rate, 2: at lowest and highest rate)
QP                  36             # Quantization parameters
MeQP0               34             # QP for mot. est. / mode decision (stage 0)
MeQP1               34             # QP for mot. est. / mode decision (stage 1)
MeQP2               34             # QP for mot. est. / mode decision (stage 2)
MeQP3               34             # QP for mot. est. / mode decision (stage 3)
MeQP4               34             # QP for mot. est. / mode decision (stage 4)
MeQP5               34             # QP for mot. est. / mode decision (stage 5)
```

Example 68: Example layer configuration file “layer1.cfg” for combined scalable coding

```
# JSVM Layer Configuration File

InputFile        SOCCER_CIF30.yuv # Input file
SourceWidth         352            # Input frame width
SourceHeight        288            # Input frame height
FrameRateIn         30             # Input frame rate [Hz]
FrameRateOut        30             # Output frame rate [Hz]
ClosedLoop          2              # Closed-loop control (0,1:at highest
#     rate, 2: at lowest and highest rate)
InterLayerPred      2              # Inter-layer Pred. (0: no, 1: yes, 2:adapt.)
QP                  36             # Quantization parameters
MeQP0               34             # QP for mot. est. / mode decision (stage 0)
MeQP1               34             # QP for mot. est. / mode decision (stage 1)
MeQP2               34             # QP for mot. est. / mode decision (stage 2)
MeQP3               34             # QP for mot. est. / mode decision (stage 3)
MeQP4               34             # QP for mot. est. / mode decision (stage 4)
MeQP5               34             # QP for mot. est. / mode decision (stage 5)
```

Example 69: Example layer configuration file “layer2.cfg” for combined scalable coding

```
# JSVM Layer Configuration File

InputFile        SOCCER_4CIF30.yuv # Input file
SourceWidth         704            # Input frame width
SourceHeight        576            # Input frame height
FrameRateIn         30             # Input frame rate [Hz]
FrameRateOut        30             # Output frame rate [Hz]
ClosedLoop          2              # Closed-loop control (0,1:at highest
#     rate, 2: at lowest and highest rate)
NumFGSLayers        1              # Number of FGS layers ( 1 layer -> DQP = 6 )
InterLayerPred      2              # Inter-layer Pred. (0: no, 1: yes, 2:adapt.)
QP                  36             # Quantization parameters
MeQP0               34             # QP for mot. est. / mode decision (stage 0)
MeQP1               34             # QP for mot. est. / mode decision (stage 1)
MeQP2               34             # QP for mot. est. / mode decision (stage 2)
MeQP3               34             # QP for mot. est. / mode decision (stage 3)
MeQP4               34             # QP for mot. est. / mode decision (stage 4)
MeQP5               34             # QP for mot. est. / mode decision (stage 5)
```
The commands that are executed for generating the combined scalable bit-stream are given in Example 70. Let’s assume the input sequence is given in 4CIF resolution (704x576 samples) at a frame rate of 30 Hz. First, downsampled sequences for the CIF and QCIF layer are generated by using the resampler. These sequences are required as input for the encoder.

The actual encoding proceeds in three steps. In the first steps, only the QCIF base layer is encoded with 1 FGS layer, and the corresponding FGS information are stored to the file “fgs0.dat”. In the next encoder call, the QCIF layer is actually encoded with a maximum bit-rate of 250 kbit/s, and additionally the CIF layer is encoded with 1 FGS layer and the FGS information are saved to the file “fgs1.dat”. In the final encoder run, all 3 layers (QCIF, CIF, and 4CIF) are encoded. For the QCIF and CIF layer, the FGS information that have been generated in the previous encoder runs are used for an encoder side truncation of corresponding PR slices. The maximum bit-rate for these layers is set to 250 and 800 kbit/s. The 4CIF layer is encoded with one complete FGS layer as specified in the corresponding layer configuration file in Example 69.

**Example 70: Example encoder call for combined scalable coding**

```bash
>DownConvertStatic 704 576 SOCCER_4CIF30.yuv 352 288 SOCCER_CIF30.yuv 0
>DownConvertStatic 352 288 SOCCER_CIF30.yuv 176 144 SOCCER_QCIF30.yuv 0

>H264AVCEncoderLibTestStatic -pf main.cfg -numl 1 -anafgs 0 1 fgs0.dat
>H264AVCEncoderLibTestStatic -pf main.cfg -numl 2 -encfgs 0 250 fgs0.dat \ 
   -anafgs 1 1 fgs1.dat
>H264AVCEncoderLibTestStatic -pf main.cfg \ 
   -encfgs 0 250 fgs0.dat \ 
   -encfgs 1 800 fgs1.dat
```

**Summary:**

<table>
<thead>
<tr>
<th>SNR Level</th>
<th>bitrate</th>
<th>Y-PSNR</th>
<th>U-PSNR</th>
<th>V-PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>176x144 @ 1.8750</td>
<td>0.0000</td>
<td>25.6995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176x144 @ 1.8750</td>
<td>1.0000</td>
<td>58.3950</td>
<td>40.4268</td>
<td>43.8784</td>
</tr>
<tr>
<td>176x144 @ 3.7500</td>
<td>0.0000</td>
<td>51.9758</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176x144 @ 3.7500</td>
<td>1.0000</td>
<td>98.1395</td>
<td>39.5453</td>
<td>43.3452</td>
</tr>
<tr>
<td>176x144 @ 7.5000</td>
<td>0.0000</td>
<td>76.4905</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176x144 @ 7.5000</td>
<td>1.0000</td>
<td>140.0558</td>
<td>38.7621</td>
<td>42.8914</td>
</tr>
<tr>
<td>176x144 @ 15.0000</td>
<td>0.0000</td>
<td>105.3168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176x144 @ 15.0000</td>
<td>1.0000</td>
<td>190.4528</td>
<td>38.1694</td>
<td>42.6757</td>
</tr>
<tr>
<td>176x144 @ 30.0000</td>
<td>0.0000</td>
<td>136.7184</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176x144 @ 30.0000</td>
<td>1.0000</td>
<td>241.6752</td>
<td>37.5991</td>
<td>42.5442</td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>0.0000</td>
<td>132.1470</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 1.8750</td>
<td>1.0000</td>
<td>241.6752</td>
<td>37.5991</td>
<td>42.5442</td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>0.0000</td>
<td>228.8637</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 3.7500</td>
<td>1.0000</td>
<td>447.0750</td>
<td>39.0826</td>
<td>43.9454</td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>0.0000</td>
<td>302.6953</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 7.5000</td>
<td>1.0000</td>
<td>645.2864</td>
<td>37.4379</td>
<td>42.5493</td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>0.0000</td>
<td>499.0074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 15.0000</td>
<td>1.0000</td>
<td>998.0148</td>
<td>37.4379</td>
<td>42.5493</td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>0.0000</td>
<td>998.0148</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352x288 @ 30.0000</td>
<td>1.0000</td>
<td>2016.6286</td>
<td>36.8233</td>
<td>42.7150</td>
</tr>
<tr>
<td>704x576 @ 1.8750</td>
<td>0.0000</td>
<td>680.8753</td>
<td></td>
<td></td>
</tr>
<tr>
<td>704x576 @ 1.8750</td>
<td>1.0000</td>
<td>1187.3163</td>
<td>38.9349</td>
<td>43.1050</td>
</tr>
<tr>
<td>704x576 @ 3.7500</td>
<td>0.0000</td>
<td>882.4405</td>
<td></td>
<td></td>
</tr>
<tr>
<td>704x576 @ 3.7500</td>
<td>1.0000</td>
<td>1517.5216</td>
<td>37.8342</td>
<td>43.7280</td>
</tr>
<tr>
<td>704x576 @ 7.5000</td>
<td>0.0000</td>
<td>1141.4016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>704x576 @ 7.5000</td>
<td>1.0000</td>
<td>2292.8032</td>
<td>37.1194</td>
<td>43.5500</td>
</tr>
<tr>
<td>704x576 @ 15.0000</td>
<td>0.0000</td>
<td>1427.4864</td>
<td></td>
<td></td>
</tr>
<tr>
<td>704x576 @ 15.0000</td>
<td>1.0000</td>
<td>2854.9728</td>
<td>36.5321</td>
<td>43.4176</td>
</tr>
</tbody>
</table>

**Example 71: Example of supported scalable layers for combined scalable coding**

```bash
>BitStreamExtractorStatic test.264

Contained Layers:
-----------------
```
The extractable bit-rates (minimum and maximum value) for each spatio-temporal resolution are specified by the encoder output at the end of Example 70 as well as by the extractor output in Example 71. The spatio-temporal resolution and the corresponding rate ranges that are supported by the generated combined scalable bit-stream are summarized in Table 5.

Table 5: Supported bit-rates for combined scalable coding

<table>
<thead>
<tr>
<th>supported bit-rates [kbit/s]</th>
<th>QCIF</th>
<th>CIF</th>
<th>4CIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>spatial resolution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QCIF</td>
<td>26-58</td>
<td>52-98</td>
<td>76-140</td>
</tr>
<tr>
<td>CIF</td>
<td>132-179</td>
<td>229-383</td>
<td>303-499</td>
</tr>
<tr>
<td>4CIF</td>
<td>285-447</td>
<td>681-1187</td>
<td>882-1518</td>
</tr>
</tbody>
</table>

The supported spatio-temporal resolutions and rate intervals can be interpreted to form a spatio-temporal-quality cube as depicted in Figure 4. Starting from a global bit-stream with the maximum spatial and temporal resolution as well as the maximum bit-rate, sub-streams can be extracted by moving through the spatio-temporal-quality cube and thus decreasing the spatial and/or temporal resolution and/or decreasing the bit-rate. In Figure 4, several possible extraction paths are illustrated.
3.6 **Fragmented NAL units and discardable sub-streams**

Fragmented NAL units are used jointly with discardable stream. In spatial scalability, discardable streams allows to define extraction points higher than the prediction point used for the upper layer. This only applies on progressive refinement slices. If the prediction rate does not correspond to a complete FGS pass, the corresponding NAL unit will be cut in two parts. The first part corresponds to the data used for prediction of the upper layer, the second part corresponds to discardable data. Each fragment is a NAL unit. In the decoding process, the fragmented NAL units are concatenated before decoding if layer 0 is decoded, otherwise only the first fragment will be decoded. To encode such a stream, the –ds option has to be used at the encoder.

**Example 72: Using Fragmented NAL units**

```bash
> H264AVCEncoderLibTestStatic.exe -pf <mcfg> -numl 1 -anafgs 0 2 fgs_0.dat
> H264AVCEncoderLibTestStatic.exe -pf <mcfg> -numl 2 -encfgs 0 r1 fgs_0.dat
   -ds 0 r2 -anafgs 1 2 fgs_1.dat
```

Example 72 illustrates the usage of the Fragmented NAL units. The first command line will generate a stream at layer 0 with the analysis of two FGS layer. Then, the second command line will generate a stream with two layers. The layer 0 will be encoded and truncated at the rate r1. The upper layer (layer 1) will be encoded with an analysis of two FGS layers and using the point of layer 0 at the rate r2 as prediction. Above the rate r2, the data of layer 0 are not used for layer 1 and can be discarded. They will not be decoded if layer 1 is decoded.

4 **Information for Software Integration**

At the JVT meetings, the integration of proposals into the JSVM 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 “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 JSVM software for their adopted tool at the next meeting.

4.1 Software integration guidelines and rules

When integrating adopted proposals into the JSVM 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 a 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 JSVM 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 JSVM version number (i.e. “.JSVM_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 JSVM 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 “changes.txt” in the root directory of the JSVM software – the changes should be directly written to the file “changes.txt”),
- An updated version of the software manual (when parameters or tools have been added).

4.3 Validation scripts

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 JSVM 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 JSVM 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 JSVM decoder. The numbering of the conformance tests corresponds to the classification of the conformance bit-streams 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 JSVM 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 to first generate the PacketLossSimulator 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 73 illustrates the dump.pm usage.

**Example 73: Using the dump.pm script**

<table>
<thead>
<tr>
<th>Usage: dump.pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-c] : to copy the &quot;conformance sequences and bitstreams&quot; in the corresponding simus directories.</td>
</tr>
<tr>
<td>[-r] : to remove the &quot;conformance sequences and bitstreams&quot; of each simus directories.</td>
</tr>
<tr>
<td>[-simu &lt;name_simu1&gt;...&lt;name_simuN&gt;] : name of the simulations to copy/remove.</td>
</tr>
<tr>
<td>[-data &lt;yuv_streams_directory&gt;] : name of the directory containing the &quot;conformance sequences and bitstreams&quot;.</td>
</tr>
<tr>
<td>[-which] : print the name of &quot;conformance sequences and bitstreams&quot; to be used.</td>
</tr>
<tr>
<td>[-u] : Usage.</td>
</tr>
</tbody>
</table>

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 74 illustrates the way to use the dump.pm script in order to know the conformance bitstreams and sequences needed when running test named 6.6.1.

**Example 74: Knowing the necessary conformance bitstreams and sequences**

```bash
> 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 75 illustrates the way the dump.pm script should be run.

**Example 75: Copying conformance data using the dump.pm script**

```bash
> 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, and DownConvertStatic.

### 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 76). By default, the Short_term test set will be run assuming ./bin as binaries directory and ./orig as YUV sequences directory.

**Example 76: Using the run.pm script**

<table>
<thead>
<tr>
<th>Usage: run.pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-SimusetName [&lt;SimuName1&gt;...&lt;SimuNameN&gt;]] : to specify that the simulations-set called &quot;SimusetName&quot; must be run (default value: Short_term)</td>
</tr>
<tr>
<td>&quot;SimusetName&quot; must be the name of a sub-directory of</td>
</tr>
</tbody>
</table>
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 test set names 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 77 illustrates the use of the run.pm script when the user intends to validate the Long_term tests set.

Additionally 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 77: Validating the Long_term tests set**

```plaintext
> perl run.pm -Long_term
is equivalent to
> perl run.pm -L
```

Example 78 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 78: Validating the tests CAVLC, ESS and T1 of the Short_term set**

```plaintext
> 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 JSVM tool. Example 79 illustrates the log file CAVLC_Global.log assuming a successful validation of the CAVLC test.

**Example 79: 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