LCEVC MPEG-5 – Part 2
TEST MODEL SOFTWARE MANUAL

(LTM 5.3)

LCEVC MPEG-5 – Part 2 Test Model Software Manual (revised for LTM 5.3)

Table of contents

List of Tables

Revision

Release No.	Date	Author	Revision Description
Rev. 0	June 2019	Lorenzo Ciccarelli	Initial version
Rev. 1	December 2019	Lorenzo Ciccarelli	LTM 5.0
Rev. 2	January 2021	Lorenzo Ciccarelli	LTM 5.0.4
Rev. 3	February 2021	Florian Maurer	LTM 5.1
Rev. 4	April 2021	Florian Maurer	LTM 5.1.2
Rev. 5	July 2021	Florian Maurer	LTM 5.3.1
Rev. 6	September 2021	Florian Maurer	LTM 5.3.2

1 Introduction

This document provides description of the Test Model of Low Complexity Enhancement Video Coding (LTM 5.3). This includes information about the encoder and decoder input parameters, syntax, compilation issues, and additional information with regards to best usage and configuration of this software.

2 Source Code and Information

The source code of the Test Model can be found in the following gitlab repository:
http://mpegx.int-evry.fr/software/MPEG/LCEVC/LTM.

A ReadMe file is provided on the same repository, containing information relative to the software implementation (LTM 5.3) and guidance on how to use it.

The configuration file for LTM 5.3 is provided by using the script stresstest.py from the directory stress_tests described in paragraph 6.

3 Installation and compilation

Prior compilation the git submodules needs to be aligned using the following command:

git submodule update --init

There are 2 ways to compile for Linux,

1. use mk.sh typing ‘sh mk.sh. and eventually ‘sh mk.sh clean’ to clean the project.
2. using cmake and typing the following commands:

```
mkdir build
cd build
cmake ..
make
```

To compile in Windows, please use cmake using the following command

```
mkdir _build_win64
cd _build_win64
cmake -G "Visual Studio 16 2019" -A x64 ..
cmake --build .
```

Please note that the base encoder or decoder (AVC, HEVC, EVC, VVC) may be called by the LTM Encoder to construct the base encoded AVC, HEVC, EVC or VVC bitstream as well as the corresponding YUV sequence. Thus, it is essential to have the `external_codecs` folder copied in the working directory.

4 Using the encoder

The encoder executable (ModelEncoder.exe [--help] [OPTION...]) can take as inputs the following:

- Source: a source video in raw YUV format, either 8 bpp, 10 bpp, 12 bpp, or 14bpp;
- Base encodes: a base format elementary stream and a reconstructed base video YUV;
- Configuration file: optionally, a configuration file for the encoder, e.g. config.json;
- A description of video format.

The base format can be either H.264/AVC, H.265/HEVC, ETM or VVC.

There are four possible modes of operation of the LTM Encoder, depending on the input sequence provided:

1. using as input an original YUV sequence, the Encoder will first invoke the "external base encoder" to produce the related base bitstream and the corresponding YUV reconstruction then, the LTM Encoder will use such base bitstream plus YUV sequence for its operation;
2. using as input an already encoded base, without the corresponding YUV, the Encoder will first invoke the related "external base decoder" to produce the corresponding YUV reconstruction then, the LTM Encoder will use such base bitstream plus YUV sequence for its operation;
3. using as input an already encoded base, with the corresponding YUV, the LTM Encoder will directly use such base bitstream plus YUV sequence for its operation.
4. As a pure downsampler and/or upscaler using LCEVC normative upscaler.

The following tables list the parameters that are available at the encoder side. In order to set Boolean parameters, the equal sign (‘=’) must be used (e.g.: --dump_surfaces=true).

In case the option --parameter_config=default is used, some of the parameter values will be derived from the input step width of enhancement sub-layer 2. To disable this feature, please use --parameter_config=conformance.

ModelEncoder [--help] [OPTION...])

Table 1 - General encoder options

Option	Default	Description
-i, --input_file	source.yuv	Input filename for raw YUV video frames
-o, --output_file	output.lvc	Output filename for elementary stream
-w, --width	1920	Image width
-h, --height	1080	Image height
-r, --fps	50	Frame rate
-l, --limit	none	Limit number of frames to encode Note: The minimum number of frames to encode depends on the group of picture structure of the base. This means that the LTM potentially needs to decode a variable minimum amount of frames depending on the base encoder whatever the value specified by the user in this parameter.
-f, --format	yuv420p	Picture format of the input source file (yuv420p, yuv420p10, yuv420p12, yuv420p14, yuv422p, yuv422p10, yuv422p12, yuv422p14, yuv444p, yuv444p10, yuv444p12, yuv444p14, y, y10, y12, or y14)
-p, --parameters	{}	JSON parameters for encoder (path to .json file)
--parameter_config	default	Configuration with default parameter values to use (default or conformance)
--dump_surfaces	false	Dump intermediate surfaces to yuv files
--output_recon	none	Output filename for encoder yuv reconstruction (must be

		specified for output)
--encapsulation	nal	Enhancement encapsulation as SEI or NAL
--version		Show version
--help		Show this help

Table 2 – Base encoder configuration options

Option	Default	Description
-b, --base_encoder	avc	Base encoder plugin to use (avc, hevc, evc or vvc)
--qp	28	QP value to be used by the specified base encoder
--base		Encoded base bitstream (if not specified the base encoder model will be invoked)
--base_recon		Decoded YUV for base bitstream (if not specified the base encoder model will be invoked or if –base is specified the only base decoder model will be invoked)
--keep_base		Keep the encoded base bitstream and reconstruction
--intra_period	1 sec	Intra Period for base encoding (default: derived from framerate)
--base_depth		Bit depth of base encoder

Table 3 – Sequence configuration options

Option	Default	Description
--profile_idc	auto	Profile (see Annex A) (auto, main, or main444) auto chooses the correct profile automatically
--level_idc	4	Level (see Annex A)
--sublevel_idc	1	Sublevel (see Annex A)
--conformance_window	false	Turn signalling of conformance cropping window offset parameters on
--extended_profile_idc	0	Extended profile (see Annex A)
--extended_level_idc	0	Extended level (see Annex A)
--conf_win_left_offset	0	Left offset of the conformance window
--conf_win_right_offset	0	Right offset of the conformance window
--conf_win_top_offset	0	Top offset of the conformance window
--conf_win_bottom_offset	0	Bottom offset of the conformance window

Table 4 – Global configuration options

Option	Default	Description
--num_image_planes	3	Number of planes in input sequence
--num_processed_planes	1	Number of planes for which enhancement shall be added (1 or 3)
--predicted_residual	true	Predicted residuals after upscaling
--encoding_transform_type	dds	Transform type (dd or dds)
	true	Temporal prediction for enhancement sub-layer 2
--temporal_use_reduced_signalling	true	Reduced signalling (tile based) for temporal
--encoding_upsample	modifiedcubic	Upsample filter
--temporal_step_width_modifier	48	Temporal step width modifier
--level_1_filtering_first_coefficient	0	L-1 filter 1st coefficient
--level_1_filtering_second_coefficient	0	L-1 filter 2nd coefficient
--scaling_mode_level1	none	Scaling mode between encoded base and preliminary intermediate picture (none, 1d or 2d)
--scaling_mode_level2	2d	Scaling mode between combined intermediate picture and preliminary output picture (none, 1d or 2d) Scaling mode between combined intermediate picture and preliminary output picture (none, 1d or 2d)
--user_data_enabled	false	User data in enhancement sub-layer 1 (none, 2bits or 6bits)
--level1_depth_flag	false	Flag for bit depth of enhancement sub-layer 1
--tile_width	0	Width of a tile (disable tiling with 0)
--tile_height	0	Height of a tile (disable tiling with 0)

--compression_type_entropy_enabled_per_tile		Use compression to signal entropy_enabled when tiling is used
--compression_type_size_per_tile	none	Compression type per tile (none, prefix or prefix_diff)

Table 5 – Picture configuration options

Option	Default	Description
--cq_step_width_loq_1	32767	Step width for enhancement sub-layer 1 (range: [4, 32767])
--cq_step_width_loq_0	32767	Step width for enhancement sub-layer 2 (range: [4, 32767])
--temporal_cq_sw_multiplier	1000	Multiplier to reduce the enhancement sub-layer 2 stepwidth of an I frame (i.e. 1000 means sub-layer 2 stepwidth value is multiplied by 1 and 500 means the sub-layer 2 stepwith value is multiplied by 0.5)
--temporal_signalling_present	true	Signal temporal layer although no_enhacement is enabled
--picture_type	frame	Picture type (frame or field)
--dithering_control	false	Dithering on
--dithering_type	none	Dithering type (none, uniform or uniform_fixed)
--dithering_strength	0	Strength of the dithering
--dequant_offset_mode	0	Dequantization offset mode (default or const_offset)
--dequant_offset	none	Offset of the dequantization
--dequant_offset_signalled	false	Dequantization offset is signalled
--quant_matrix_mode	previous	Quantization mode (previous, default, custom, custom_default, default_custom or custom_custom)
--qm_coefficient_1		Custom quantization coefficients for sub-layer 1 (must be 16 values for 4x4 transform and 4 for 2x2 trasform) The form of input has to be of the form i.e. of 2x2 transform --qm_coefficient_1=" 0 0 0 0"
--qm_coefficient_2		Custom quantization coefficients for sub-layer 2 (must be 16 values for 4x4 transform and 4 for 2x2 trasform) The form of input has to be of the form i.e. of 2x2 transform --qm_coefficient_2=" 0 0 0 0".
--level_1_filtering_enabled	false	Apply L-1 deblocking filter

Table 6 – Encoder configuration option

Option	Default	Description
--encoding_downsample_luma	lanczos3	Downsample filter for luma plane used to generate the luma palne of the base
--encoding_downsample_chroma	lanczos3	Downsample filter for chrominance planes of the base
--priority_mode	mode_3_1	Residual priority map running mode
--user_data_method	zeros	Type of user data to be inserted (zeros, ones, random or fixed_random) used for testing
--downsample_only		Downsample input and wrtite to output
--upsample_only		Upsample input and write to output

4.1 Examples of using the encoder

4.1.1 Encode an 8 bit LCEVC HD stream, based on AVC

ModelEncoder --width=1920 --height=1080 --format=yuv420p --input_file=Cact_1920x1080_50fps_08bpp.yuv --output_file=Cact_1920x1080_50fps_QP24_1000.lvc --output_recon=Cact_1920x1080_50fps_QP24_1000_enc.yuv --base_encoder=avc --encapsulation=nal --base=Cact_0960x0540_50fps_QP24.avc --base_recon=Cact_0960x0540_50fps_QP24_avc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.2 Encode a 10 bit LCEVC HD stream, based on AVC

ModelEncoder --width=1920 --height=1080 --format=yuv420p10 --input_file=Cact_1920x1080_50fps_10bpp.yuv --output_file=Cact_1920x1080_50fps_QP24_1000.lvc --output_recon=Cact_1920x1080_50fps_QP24_1000_enc.yuv --base_encoder=avc --encapsulation=nal --base=Cact_0960x0540_50fps_QP24.avc --base_recon=Cact_0960x0540_50fps_QP24_avc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.3 Encode an 8 bit LCEVC UHD stream, based on HEVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p --input_file=Park_3840x2160_50fps_08bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=hevc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_hevc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.4 Encode a 10 bit LCEVC UHD stream, based on HEVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p10 --input_file=Park_3840x2160_50fps_10bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=hevc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_hevc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.5 Encode an 8 bit LCEVC UHD stream, based on EVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p --input_file=Park_3840x2160_50fps_08bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=evc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_evc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.6 Encode a 10 bit LCEVC UHD stream, based on EVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p10 --input_file=Park_3840x2160_50fps_10bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=evc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_evc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.7 Encode an 8 bit LCEVC UHD stream, based on VVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p --input_file=Park_3840x2160_50fps_08bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=vvc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_vvc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.8 Encode a 10 bit LCEVC UHD stream, based on VVC

ModelEncoder --width=3840 --height=2160 --format=yuv420p10 --input_file=Park_3840x2160_50fps_10bpp.yuv --output_file=Park_3840x2160_50fps_QP24_1000.lvc --output_recon=Park_3840x2160_50fps_QP24_1000_enc.yuv --base_encoder=vvc --encapsulation=nal --base=Park_1920x1080_50fps_QP24.hevc --base_recon=Park_1920x1080_50fps_QP24_vvc.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.9 Encode a 10 bit LCEVC UHD stream, based on an 8 bit base encoder (AVC)

ModelEncoder --width=1920 --height=1080 **--format=yuv420p10 –base_debpth=8** --input_file=Cact-1920x1080_50fps_10bpp.yuv --output_file=Cact_1920x1080_50fps_QP24_1000_10bpp.lvc --output_recon=Cact_1920x1080_50fps_QP24_1000_enc_10bpp.yuv --base_encoder=avc --encapsulation=nal --base=Cact_0960x0540_50fps_QP24_8bbp.avc --base_recon=Cact_0960x0540_50fps_QP24_avc_8bpp.yuv --limit=500 --cq_step_width_loq_1=32767 --cq_step_width_loq_0=1000

4.1.10 Encoder as downsampler only

8 bit input
ModelEncoder --downsample_only=true --format=yuv420p -w 1920 -h 1080 -i in_1920x1080.yuv -o down_960x540.yuv

10 bit input
ModelEncoder --downsample_only=true --format=yuv420p10 -w 1920 -h 1080 -i in_1920x1080.yuv -o down_960x540.yuv

4.1.11 Encoder as upsampler only

8 bit input
ModelEncoder --upsample_only=true --format=yuv420p -w 1920 -h 1080 -i down_960x540.yuv -o up_1920x1080.yuv

10 bit input
ModelEncoder --upsample_only=true --format=yuv420p10 -w 1920 -h 1080 -i down_960x540.yuv -o up_1920x1080.yuv

Note: Width and height are always for the enhanced image size, so input size for downsample, output size for upsample.

5 Using the decoder

The decoder executable ModelDecoder (ModelDecoder.exe in Windows) can take as inputs the following:

- An LCEVC bitstream (.lvc),
- The type of codec which was used to encode the base (H.264/AVC, H.265/HEVC, EVC, VVC)
- The type of encapsulation (NAL or SEI)

The following tables list the parameters that are available at the decoder side. In order to set those parameters, the sign = must be used (e.g.: --dump_surfaces=true).

ModelDecoder [--help] [OPTION...])

Table 7 - General decoder options

Option	Default	Description
-i, --input_file	input.lvc	Input filename for raw YUV video frames
-o, --output_file	output.yuv	Output filename for elementary stream
-b, --base	avc	Base codec (avc, hevc, evc, vvc or yuv)
--base_encoder	avc	Base codec (same as --base)
--base_external	false	Use an external base codec executable
-y, --base_yuv		Prepared YUV data for base decode
--dump_surfaces	false	Dump intermediate surfaces to yuv files
--encapsulation	nal	Wrap enhancement as SEI or NAL (nal or sei)
--dithering_switch	true	If set to false, the dithering will not be added to the final reconstructed picture even if transmitted.
--dithering_fixed	false	Use a fixed seed for dithering
--keep_base	false	Save the decoded base yuv file as well as the demultiplexed base and enhancement bitstreams
--version		Show version
--help		Show this help

The LTM decoder can be used to extract an enhancement bitstream from a multiplexed bitstream. Additionally, the decoding operation of the base bitstream is performed to achieve the decoded base pictures.

Assuming a multiplexed bitstream input.bit as input, the following example command line can be used:

ModelDecoder.exe -i str.bit -b avc --keep_base=true

The outputs of this operation are a YUV video file containing the decoded base pictures, the base bitstream, and the enhancement bitstream.

5.1 Examples of using the decoder

5.1.1 Decode an 8 bit LCEVC HD stream, based on AVC

ModelDecoder --base_encoder=avc --input_file=Cact_1920x1080_50fps_08bpp.lvc --output_file=Cact_1920x1080_50fps_08bpp_dec.yuv

5.1.2 Decode a 10 bit LCEVC HD stream, based on AVC

ModelDecoder --base_encoder=avc --input_file=Cact_1920x1080_50fps_10bpp.lvc --output_file=Cact_1920x1080_50fps_10bpp_dec.yuv

5.1.3 Decode an 8 bit LCEVC UHD stream, based on HEVC

ModelDecoder --base_encoder=hevc --input_file=Park_3840x2160_50fps_08bpp.lvc --output_file=Park_3840x2160_50fps_08bpp_dec.yuv

5.1.4 Decode a 10 bit LCEVC UHD stream, based on HEVC

ModelDecoder --base_encoder=hevc --input_file=Park_3840x2160_50fps_10bpp.lvc --output_file=Park_3840x2160_50fps_10bpp_dec.yuv

5.1.5 Decode an 8 bit LCEVC UHD stream, based on EVC

ModelDecoder --base_encoder=evc --input_file=Park_3840x2160_50fps_08bpp.lvc --output_file=Park_3840x2160_50fps_08bpp_dec.yuv

5.1.6 Decode a 10 bit LCEVC UHD stream, based on EVC

ModelDecoder --base_encoder=evc --input_file=Park_3840x2160_50fps_10bpp.lvc --output_file=Park_3840x2160_50fps_10bpp_dec.yuv

5.1.7 Decode an 8 bit LCEVC UHD stream, based on VVC

ModelDecoder --base_encoder=vvc --input_file=Park_3840x2160_50fps_08bpp.lvc --output_file=Park_3840x2160_50fps_08bpp_dec.yuv

5.1.8 Decode a 10 bit LCEVC UHD stream, based on VVC

ModelDecoder --base_encoder=vvc --input_file=Park_3840x2160_50fps_10bpp.lvc --output_file=Park_3840x2160_50fps_10bpp_dec.yuv

5.1.9 Decode an 8 bit LCEVC HD stream with base YUV file (no base codec)

ModelDecoder --base_encoder=yuv --input_file=Cact_1920x1080_50fps_08bpp.lvc --output_file=Cact_1920x1080_50fps_08bpp_dec.yuv --base_yuv Cact_960x540_50fps_08bpp.yuv

6 LTM test configuration generator

Along with the code the project provides a script to generate the configuration file to run smoke tests of the LTM. The script is called *stresstest.py* and it is located under the project root directory */stress_tests.* Following the command line help is described:

```
stresstest.py
        [--encoder=<encoder>]
        [--decoder=<decoder>]
        [--test_file=<tests-json>]
        [--checksums_file=<checksums-json>]
        [--label=<label>]
        [--input_dir=<dir>]
        [--base_dir=<dir>]
        [--progress=none|spinner|verbose]
        [--sets=<string>,..]
        [--single=<string>]
        [--parallel=<number>]
        [--generate_checksums_file=<file>]
```

The stress test script will generate the LTM configuration file which is described by a JSON file with the following content:

```
[ // Array of tests
  {
    "description" : <string>  // optional, human readable description
    "name" : <string>         // identifier used to identity directory and files generated by test
    "include-parameters" : <filename> // optional, start with parameters in named file then apply 'paramaters'
    "parameters" : { ... }    // json dictionary of codec parameters
    "sets" : [ <strings> ]    // optional, list  of sets that include this test - controller by  command line --sets=...
    "generate_base" : <bool>  // optional, control whether base+recon should be encoded for this test, overriding command line
  }
  ...
]
```

Some parameters are, however, treated specially:

```
- input_file,  width, height, fps, bit_depth and format
  are extracted from filename if present (but are overridden by explicit params)

- input_file, base,  base_recon: Input directory is prepended
```

In the same directory an example 'tests.json' is provided that exercises all the codec tools.

Examples of applying it to sequences:

```
HD AVC:
stresstest.py --progress=spinner  --label="Cactus_AVC_" --test_file=tests.json --parameters='
{
   "input_file":"Cactus_49frames_1920x1080_50fps_420.yuv",
   "base":"Cactus_49frames_base_avc_qp19_960x540_50fps.bin",
   "base_recon":"Cactus_49frames_base_avc_qp19_960x540_50fps.yuv",
   "format":"yuv420p",
   "base_encoder":"avc"
}
'
UHD HEVC:
./stresstest.py  --progress=spinner --label="ParkRunning3_HEVC_" --test_file=tests.json --parameters='
{
  "input_file":"ParkRunning3_63frames_3840x2160_50fps_10bit_420.yuv",
  "base":"ParkRunning3_63frames_base_hevc_qp19_1920x1080.bin",
  "base_recon":"ParkRunning3_63frames_base_hevc_qp19_1920x1080.yuv",
  "format":"yuv420p10",
  "base_encoder":"hevc"
}
```

7 References

[1] “Draft Text of ISO/IEC DIS 23094-2, Low Complexity Enhancement Video Coding”, ISO/IEC JTC1/SC29 WG11 Doc. w18986, Brussels, BE, January 2020.

[2] “Common Test Condition of Low Complexity Enhancement Video Coding”, ISO/IEC JTC1/SC29 WG11 Doc. w18988, Brussels, BE, January 2020.