Git

**ISO/IEC JTC 1/SC 29/WG 11**

**Coding of moving pictures and audio**

**Convenorship: UNI (Italy)**

**ISO/IEC JTC 1/SC 29/WG 11 N 19237**

**Document type: Approved WG 11 document**

**Title: Test Model 4.1 of Low Complexity Enhancement Video Coding**

**Status: Approved**

**Date of document: 2020-06-03**

**Source: Video**

**Expected action: N/A**

**No. of pages: 13**

**Email of convenor: leonardo@chiariglione.org**

**Committee URL: mpeg.chiariglione.org**

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 11 N18987**

**Alpbach, AT - April 2020**

|  |  |
| --- | --- |
| **Source:** | **Video** |
| **Title:** | **Test Model 4 of Low Complexity Enhancement Video Coding** |
| **Authors:** | **Simone Ferrara (V-Nova), Stefano Battista (Universita’ Politecnica delle Marche), Florian Maurer (RWTH Aachen University), Lorenzo Ciccarelli (V-Nova)** |

Contents

[1 Introduction 3](#_Toc40824822)

[2 General description of the algorithm 3](#_Toc40824823)

[2.1 Encoding process 3](#_Toc40824824)

[2.2 Decoding process 4](#_Toc40824825)

[3 Description of encoder tools 5](#_Toc40824826)

[3.1 Downsampling 5](#_Toc40824827)

[3.2 Residual encoding 5](#_Toc40824828)

[3.2.1 Transform 5](#_Toc40824829)

[3.2.2 Quantization 7](#_Toc40824830)

[3.2.3 Entropy coding 7](#_Toc40824831)

[3.3 Residual decoding 7](#_Toc40824832)

[3.4 L-1 filter 7](#_Toc40824833)

[3.5 Upsampling 7](#_Toc40824834)

[3.6 Predicted Residuals 7](#_Toc40824835)

[3.7 Temporal prediction 7](#_Toc40824836)

[3.8 Level-2 (L-2) decoding 8](#_Toc40824837)

[4 Source Code and Information 8](#_Toc40824838)

[5 Installation and compilation 8](#_Toc40824839)

[6 Using the encoder 8](#_Toc40824840)

[6.1 Example of using the encoder 11](#_Toc40824841)

[6.1.1 Encode an 8 bit HD stream, based on AVC 11](#_Toc40824842)

[6.1.2 Encode a 10 bit HD stream, based on AVC 11](#_Toc40824843)

[6.1.3 Encode an 8 bit UHD stream, based on HEVC 11](#_Toc40824844)

[6.1.4 Encode a 10 bit UHD stream, based on HEVC 12](#_Toc40824845)

[6.1.5 Encoder as upsampler only 12](#_Toc40824846)

[7 Using the decoder 12](#_Toc40824847)

[7.1 Example of using the decoder 13](#_Toc40824848)

[7.1.1 Decode an 8 bit HD stream, based on AVC 13](#_Toc40824849)

[7.1.2 Decode a 10 bit HD stream, based on AVC 13](#_Toc40824850)

[7.1.3 Decode an 8 bit UHD stream, based on HEVC 13](#_Toc40824851)

[7.1.4 Decode a 10 bit UHD stream, based on HEVC 13](#_Toc40824852)

[8 References 13](#_Toc40824853)

**List of Tables**

[Table 1 - General encoder options 9](#_Toc40825022)

[Table 2 – Base encoder configuration options 9](#_Toc40825023)

[Table 3 – Sequence configuration options 10](#_Toc40825024)

[Table 4 – Global configuration options 10](#_Toc40825025)

[Table 5 – Picture configuration options 10](#_Toc40825026)

[Table 6 – Encoder configuration option 11](#_Toc40825027)

[Table 7 - General decoder options 12](#_Toc40825028)

**Test Model 4 of Low Complexity Enhancement Video Coding**

# Introduction

This document provides description of the Test Model of Low Complexity Enhancement Video Coding (LTM 4.1). Brief description of design principles; utilized algorithm and software organization are provided.

# General description of the algorithm

The general structure of the encoding scheme uses a downsampled source signal encoded with a base codec, adds a first level of correction data to the decoded output of the base codec to generate a corrected picture, and then adds a further level of enhancement data to an upsampled version of the corrected picture.

The codec format uses a minimum number of simple coding tools. When combined synergistically, they can provide visual quality improvements when compared with a full resolution picture encoded with the base codec whilst at the same time generating flexibility in the way they can be used.

This document describes the encoding and decoding processes by assuming a 2-dimensional (2D) downscaling/upscaling between enhancement sub-layers 1 and 2 and no scaling between the reconstructed base codec video and enhancement sub-layer 1. Any other combination of the scaling options (1-dimensional, 2-dimensional or no scaling) can be used for the two available up/downscaler as well.

## Encoding process

The encoding process is depicted in the block diagram in Fig. 1. An input full resolution video is processed to generate various encoded streams. The following streams are produced by the encoder:

* a first encoded stream *(Encoded Base)* is produced by feeding a base codec (e.g., AVC, HEVC, or any other codec) with a downsampled version of the input video;
* a second encoded stream *(L-1 Coefficient Layers)* is produced by processing the residuals obtained by taking the difference between the reconstructed base codec video and the downsampled version of the input video (L-1 residuals);
* a third encoded stream *(L-2 Coefficient Layers)* is produced by processing the residuals obtained by taking the difference between an upsampled version of a corrected version of the reconstructed base coded video and the input video (L-2 residuals);
* a fourth encoded stream *(Temporal Layer)* is produced from the temporal processing on enhancement sub-layer 2 to instruct the decoder and
* a fifth stream *(Headers)* are produced for configuring the decoder.



Fig. 1 – Block diagram of encoding process

## Decoding process

The decoding process is depicted in the block diagram in Fig. 2. The decoder receives the five streams generated by the encoder. These streams are processed as described in the following:

* The headers are used to derive the decoder configuration. This includes settings for the decoding tools as well as the format of the output video.
* The encoded base stream (Encoded Base) is decoded by a base decoder corresponding to the base codec used in the encoder.
* This output is combined with the decoded residuals obtained from the L-1 Coefficient Layers.
* The combined video is upsampled and further combined with the decoded residuals obtained from the L-2 Coefficient Layers.
* Finally, residuals from the previous frame will be added according to a temporal mask as signaled in the Temporal Layer.



Fig. 2 – Block diagram of decoding process

# Description of encoder tools

## Downsampling

The downsampling process is applied to the input video to produce a downsampled video to be encoded by a base codec. The downsampling can be done either in both vertical and horizontal directions, only in the horizontal direction or alternatively cannot be applied.

## Residual encoding

The following paragraphs describe the steps required for encoding residuals.

As discussed in Section 2.1, there are two types of residuals. The L-1 residuals are obtained by taking a difference between the reconstructed base codec video and the downsampled version of the input video. The L-2 residuals are obtained by difference between an upsampled version of a corrected version of the reconstructed base coded video and the input video.

The same steps apply to both L-1 and L-2 residuals.

### Transform

There are two types of transforms that can be used in the encoding process. Both of these transforms leverage small kernels which are applied directly to the residuals. Fig. 3 shows a representation of how the residuals are transformed.

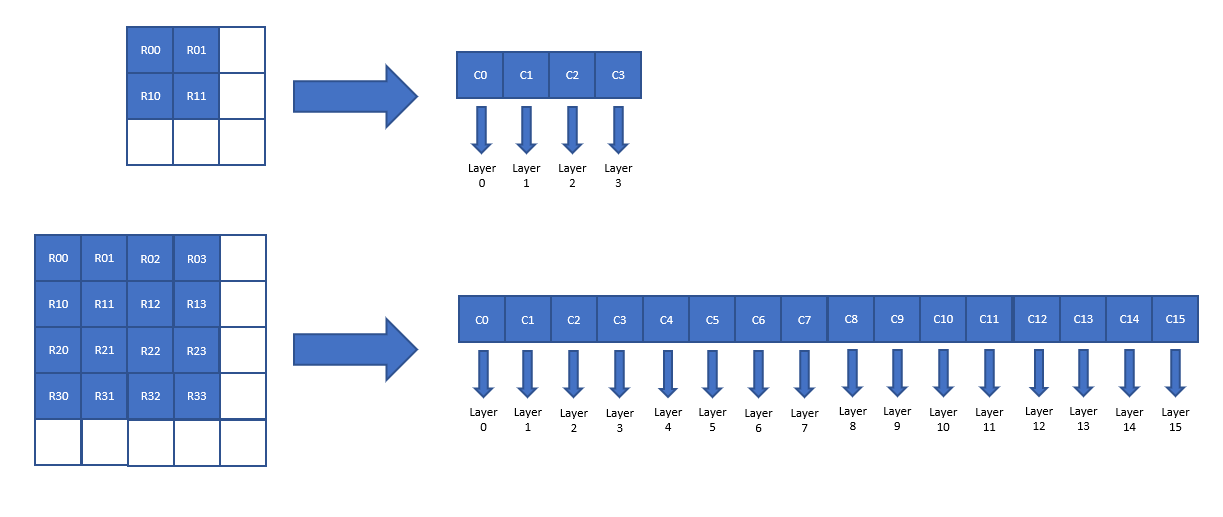


Fig. 3 – Schematic representation of transformation of residuals into coefficients and subsequent parsing into layers (called “coefficient groups” in [1]) (following quantization and entropy encoding). Top: 2x2 case, bottom: 4x4 case.

A first transform has a 2x2 kernel which is applied to each 2x2 block of residuals. The resulting coefficients are as follows:

=

A second transform has a 4x4 kernel which is applied to a 4x4 block of residuals. The resulting coefficients are as follows:

=

In the event the downsampling/upsampling is done in one direction only, the transforms are modified as follows:

=

=

### Quantization

The coefficients are then quantized using a quantizer. The quantizer may use a dead zone whose size changes relative to the quantization step.

### Entropy coding

The quantized coefficients are encoded using an entropy coder. In a scheme of entropy coding, the quantized coefficients are first encoded using run length encoding (RLE), then the encoded output is processed using a Prefix Code Encoder. There is also an option to encode using only RLE.

## Residual decoding

The input to this tool comprises the L-1 encoded residuals, which are passed through an entropy decoder, a de-quantizer and an inverse transform module. The operations performed by the entropy decoder are the inverse operations performed by the modules described in 3.2.1 - 3.2.3.

## L-1 filter

If a 4x4 transform is used, the decoded residuals may be fed to a filter module. In an implementation, the filter operates on each block of transformed residuals by applying a mask whose weights can be specified.

## Upsampling

The combination of the reconstructed L-1 residuals and base decoded video is upsampled in order to generate an upsampled reconstructed video. The upsampling method is selectable and signaled in the bytestream. Currently, there are four different upsampler kernels that could be selected.

## Predicted Residuals

Based on the Predicted Residual mode that was selected, the encoder will further modify the upsampled reconstructed video by applying Predicted Residuals (PRs). Predicted Residuals are based on the upsampled reconstructed video and on the pre-upsampling reconstructed lower resolution video.

## Temporal prediction

The temporal prediction uses a zero-motion vector prediction with a temporal buffer which stores residuals from the previous frame only.

The temporal prediction determines whether a block should be intra or inter coded. In particular, this is derived using an initial estimate provided via a temporal mask of whether (a) a tile (32x32 residuals) should be considered in its entirety an intra tile and, if not, (b) a transform block (2x2 or 4x4) should be considered an inter or intra block.

Specifically, the prediction verifies whether the number of inter blocks within a tile is above a certain threshold, in which case it decides to encode the whole tile as inter and refreshes the buffer. If that is not the case, if the tile is considered to be inter, then if the buffer is empty it encodes the whole tile as inter, if the buffer is not empty it encodes the whole tile as intra. On the other hand, if the tile is considered to be intra, it then runs a cost function to determine whether it is intra or inter.

## Level-2 (L-2) decoding

The input to this tool comprises the encoded L-2 residuals. The decoding process of the L-2 residuals are passed through an entropy decoder, a de-quantizer and an inverse transform module. The operations performed by these modules are the inverse operations performed by the modules described in 3.2.1 - 3.2.3. If the temporal selection mode has been selected, the residuals may be in part predicted from co-located residuals from a temporal buffer.

# 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 4.1.0) and guidance on how to use it.

The configuration file for LTM 4.1.0 is provided in ftp:[mpegcontent@mpegfs.int-evry.fr](mailto:mpegcontent@mpegfs.int-evry.fr)  
/MPEG-05/Part02/testbench/AdditionalTests.

# Installation and compilation

To compile for Linux, use mk.sh typing ‘sh mk.sh. and eventually ‘sh mk.sh clean’ to clean the project.

To compile in Windows, use the LCEVC\_TestModel.sln file under VisualStudio2017.

After having generated the artefacts copy the contents of the `linux` or `windows` directory as appropriate and add that directory to the search path (`PATH`).

Please note that the base encoder or decoder (AVC or HEVC) may be called by the LTM Encoder to construct the base encoded AVC or HEVC bitstream as well as the corresponding YUV sequence.<br>

Thus, it is essential to have the `external\_codecs` folder copied in the working directory.

# Using the encoder

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

- Source: a source video in raw YUV 420 format, either 8 bpp or 10 bpp;

- 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 or H.265/HEVC.

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

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

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 |
| -f, --format | yuv420p | Picture format (yuv420p or yuv420p10) |
| -p, --parameters | {} | JSON parameters for encoder (path to .json file) |
| --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 or hevc) |
| --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 | main | Profile (see Annex A) (main or main444) |
| --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 | 48 | Dithering type (none or uniform) |
| --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”. |

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\_1\_0 | Residual priority map running mode |
| --user\_data\_method | zeros | Type of user data to be inserted (zeros, ones or random) used for testing |
| --downsample\_only |  | Downsample input and wrtite to output |
| --upsample\_only |  | Upsample input and write to output |

## Examples of using the encoder

### 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

### 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

### 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

### 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

### 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

### 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.

# Using the decoder

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

- An LCEVC bitstream (.lvc),

- A description of video format,

- 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).

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 |
| -w, --width | 1920 | Image width |
| -h, --height | 1080 | Image height |
| -f, --format | yuv420p | Picture format (yuv420p or yuv420p10) |
| -b, --base | avc | Base codec (avc or hevc) |
| --base\_encoder | avc | Base codec (same as base) |
| -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 (need to be specified) |
| --version |  | Show version |
| --help |  | Show this help |

## Examples of using the decoder

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

ModelDecoder --width=1920 --height=1080 --format=yuv420p --base\_encoder=avc --input\_file=Cact\_1920x1080\_50fps\_08bpp.lvc --output\_file=Cact\_1920x1080\_50fps\_08bpp\_dec.yuv

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

ModelDecoder --width=1920 --height=1080 --format=yuv420p10 --base\_encoder=avc --input\_file=Cact\_1920x1080\_50fps\_10bpp.lvc --output\_file=Cact\_1920x1080\_50fps\_10bpp\_dec.yuv

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

ModelDecoder --width=3840 --height=2160 --format=yuv420p --base\_encoder=hevc --input\_file=Park\_3840x2160\_50fps\_08bpp.lvc --output\_file=Park\_3840x2160\_50fps\_08bpp\_dec.yuv

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

ModelDecoder --width=3840 --height=2160 --format=yuv420p10 --base\_encoder=hevc --input\_file=Park\_3840x2160\_50fps\_10bpp.lvc --output\_file=Park\_3840x2160\_50fps\_10bpp\_dec.yuv

# 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.