 ISO/IEC JTC 1/SC 29/WG 04 N0716

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

**Document type:** Output Document

**Title:** Software manual of LVTM

**Status:** Approved

**Date of document:** 2025-07-25

**Source:** ISO/IEC JTC 1/SC 29/WG 04

**Expected action:** None

**Action due date:** None

**No. of pages:** 9 (without cover page)

**Email of Convenor:** yul@zju.edu.cn

**Committee URL:** <https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-4>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 04 MPEG VIDEO CODING**

**ISO/IEC JTC 1/SC 29/WG 04 N0716**

**July 2025, Daejeon, Korea**

|  |  |
| --- | --- |
| **Title** | **Software manual of LVTM** |
| **Source** | **WG 04, MPEG Video Coding** |
| **Status** | **Approved** |
| **Serial Number** | **25494** |
| **Authors** | **Lili Zhao (China Mobile)** |

**Abstract**

This document describes the software manual for the Lenslet Video Test Model (LVTM) for the Lenslet Video Coding (LVC) project. This software description covers from the installation instruction of the LVTM package to the performance evaluation.

**Contents**

[**1** **Introduction** 3](#_Toc209323974)

[**2** **Installation and compilation** 3](#_Toc209323975)

[2.1 Dependencies 4](#_Toc209323976)

[2.2 Build instruction for LVTM 4](#_Toc209323977)

[**3** **Encoder/decoder parameters** 5](#_Toc209323978)

[**4** **Script parameters** 6](#_Toc209323979)

[**5** **Running the LVTM** 6](#_Toc209323980)

[5.1 Running the LVTM encoder 7](#_Toc209323981)

[5.2 Running the LVTM decoder 7](#_Toc209323982)

[5.3 Running the Rendering 7](#_Toc209323983)

[References 8](#_Toc209323984)

1. **Introduction**

The test model is being made available to provide a reference implementation of the LVC standard being developed by experts from ISO/IEC JTC1 SC29 WG4. One main goal of the test model is to provide a basis for experiments. These experiments help evaluate which coding tools achieve the desired performance. It is not meant to be a particularly efficient implementation of anything, and one may notice its apparent unsuitability for a particular use. It should not be construed to be a reflection of how complex a production-quality implementation of a future LVC standard would be.

This document aims to provide guidance on the usage of the LVTM software. It is widely suspected to be incomplete, and suggestions for improvements are welcome. Such suggestions and general inquiries may be sent to the coordinator (Lili Zhao, zhaoliliyjy@chinamobile.com).

**Bug reporting**

Bugs should be reported on issue tracker set up at: <https://git.mpeg.expert/MPEG/Video/lvc/rs/lvtm/-/issues>

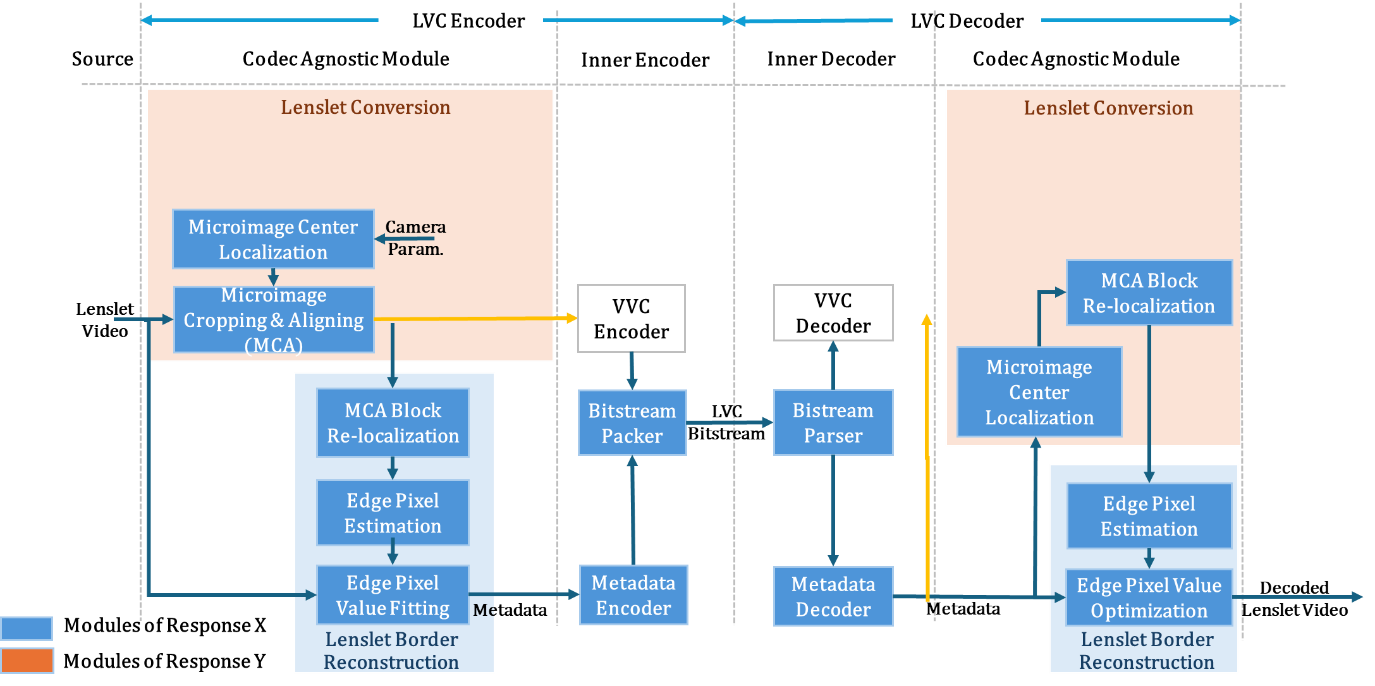


Figure 1. Architecture for the first version of LVTM

The first version of the LVTM is built on Response X and has been released. Additional modules from Response Y are under performance verification as Core Experiments, as described in [1]. Figure 1 shows the first version of the LVTM (Lenslet Video Test Model). The CTC is defined in [2]. For details of the Response X and the Response Y, refer to [1].

1. **Installation and compilation**

The software may be retrieved from the GitLab server located at <https://git.mpeg.expert/MPEG/Video/lvc/rs/lvtm>.

Table 1 lists the compiler environments and versions for which building the software is tested.

Note that the software makes use of C++20 language features, which may not be available in older compilers.

Table 1: Supported compilers

|  |  |
| --- | --- |
| **Compiler environment** | **Version** |
| Ubuntu | 22.04.5 LTS |
| Cmake | >=3.16 |
| C++ | 20 or newer compiler toolchain |
| Python | 3.10 or newer |

The software uses CMake to create platform-specific build files.

## 2.1 Dependencies

* **Zlib**

sudo apt-get install liblzma-dev zlib1g-dev

* **VVC software - VTM-11.0**

git clone https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\_VTM.git --branch VTM-11.0

Open a command prompt on your system and change into the root directory of this project.

Create a build directory in the root directory:

mkdir build

For generating Linux Release Makefile:

|  |
| --- |
| cd build  cmake .. -DCMAKE\_BUILD\_TYPE=Release |

For generating Linux Debug Makefile:

|  |
| --- |
| cd build  cmake .. -DCMAKE\_BUILD\_TYPE=Debug |

Then type

make -j

## 2.2 Build instruction for LVTM

git clone --branch LVTM https://github.com/LilydotEE/LVC\_MCA\_VVC.git LVTM

Note：A working CMake installation is required for building the software.

CMake generates configuration files for the compiler environment on each platform. The following is a list of examples for Linux (make).

Open a command prompt on your system and change into the root directory of this project.

Create a build directory in the root directory:

|  |
| --- |
| mkdir build |

Use the following CMake commands, based on your platform. Feel free to change the commands to satisfy your needs.

Linux

For generating Linux Makefile:

|  |
| --- |
| cd build  cmake .. |

Then type

|  |
| --- |
| make -j12 |

to build the software.

For more details, refer to the CMake documentation: <https://cmake.org/cmake/help/latest>

1. **Encoder/decoder parameters**

Configuration files for CTC conditions are available under *config/* directory of the provided LVTM. For each sequence, the configuration files contain:

* *<sequence>\_enc.cfg*: MCA encoder configuration
* *<sequence>\_vvc.cfg*: VTM configuration
* *<sequence>\_dec.cfg*: MCA decoder configuration

Table 2: Description of the encoder parameters in *<sequence>\_enc.cfg*

|  |  |
| --- | --- |
| **Option** | **Description** |
| InputFile | Path to the original lenslet video, usually in raw YUV. |
| OutputFile | Path to the generated data after MCA preprocessing, typically in YUV format. |
| MetaDataFile | Path to the metadata file generated during MCA preprocessing, which contains layout, width, height, and other information. |
| FramesToBeEncoded | Number of the frames to be encoded. |
| Type | The camera type: Raytrix (0) and TSPC(1), which is set according to Table 3. |
| SourceWidth | Specifies the width and height of the input lenslet video in luma samples. |
| SourceHeight |
| mode | 0, for MCA preprocessing (encoder) |
| patch | The size of the MCA block, in pixels. |
| vectors | [x, y], The prediction vector of MCA, in pixels. |
| optimize | 1 for linear optimization in *Edge Pixel Estimation and Edge Pixel Value Fitting* modules, 0 for no optimization |
| diameter | The diameter of MI, in pixels. |
| rotation | The rotation angle of MI, in radians. |
| ltop | The center of MI in top-left corner, in pixels. |
| rtop | The center of MI in top-right corner, in pixels. |
| lbot | The center of MI in bottom-left corner, in pixels. |
| rbot | The center of MI in bottom-right corner, in pixels. |

Table 3: Description of the *Type* setting

|  |  |
| --- | --- |
| **Test material filename** | **Type** |
| Origami | 0 |
| Fujita2 | 0 |
| TempleBoatGiantR32 | 0 |
| Boxer-IrishMan-Gladiator2 | 0 |
| Boys2 | 1 |
| Matryoshka | 1 |
| Motherboard2 | 1 |
| HandTools | 1 |
| MiniGarden2 | 1 |
| Origami | 0 |
| Fujita2 | 0 |
| TempleBoatGiantR32 | 0 |

For *<sequence>\_enc.cfg*, the following information should be modified based on your platform:

* **InputFile**
* **OutputFile**
* **MetaDataFile**
* **FramesToBeEncoded**

For *<sequence>\_vvc.cfg*, the following information should be modified based on your platform:

* **FramesToBeEncoded**: Number of the frames to be encoded by the VTM encoder, which should be the same with that of *<sequence>\_enc.cfg*.

Table 4: Description of the decoder parameters in *<sequence>\_dec.cfg*

|  |  |
| --- | --- |
| **Option** | **Description** |
| InputFile | Path to the yuv decoded by the VTM decoder. |
| OutputFile | Path to the reconstructed data after MCA postprocessing, typically in YUV format. |
| MetaDataFile | Path to the reconstructed metadata file, which contains layout, width, height, and other information. |
| FramesToBeEncoded | Number of the frames to be decoded, which should be the same with that of *Boys2\_enc.cfg*. |
| Type | The camera type: Raytrix (0) and TSPC(1), where is it set according to the following Table 3. |
| SourceWidth | Specifies the width and height of the input video. |
| SourceHeight |
| mode | 1, for MCA postprocessing (decoder) |

For *<sequence>\_dec.cfg*, the following information should be modified based on your platform:

* **InputFile**
* **OutputFile**
* **MetaDataFile**
* **FramesToBeEncoded**

1. **Script parameters**

* **MCA\_BIN\_PATH**: Path to the MCA encoder/decoder executable.
* **VVC\_ENCODER\_BIN\_PATH**: Path to the VVC encoder executable, used for compressing the output from MCA encoding.
* **VVC\_DECODER\_BIN\_PATH**: Path to the VVC decoder executable, used for decompressing VVC bitstreams.
* **MCA\_VVC\_MERGER\_BIN\_PATH**: Path to the merger tool executable, used for merging MCA and VVC streams or related post-processing.
* **root\_dir**: Path to the directory containing all configuration files for encoding and decoding (*config/*).
* **vtm\_cfg**: Path to the VTM encoder configuration file, located under the configuration directory (*config/encoder\_randomaccess\_vtm.cfg*).
* **data\_dir**: Path to the output of the encoder, which is same as **OutputFile** in *Boys2\_enc.cfg*.

1. **Running the LVTM**

The advice is to use the script (*scripts/run.sh*) instead of manually running encoding and decoding steps.

## 5.1 Running the LVTM encoder

Take the sequence "*Boys2\_3976x2956\_300frames\_8bit\_yuv420.yuv*" as an example, while other sequences follow the same rule.

Keep the following content in the script:

|  |
| --- |
| # encoding  python run\_lvtm.py \  --mca\_config ${root\_dir}/Boys2/Boys2\_enc.cfg \  --vvc\_vtm\_config ${vtm\_cfg} \  --vvc\_data\_config ${root\_dir}/Boys2/Boys2\_vvc.cfg \  --qp 46 |

Then

bash run.sh

In this example, the following files will be produced after encoding:

* Boys2.yuv (output of MCA preprocessing, as the input of the VTM encoder)
* Boys2.log (output of MCA preprocessing)
* Boys2\_vvc.bin (VVC bitstreams)
* Boys2\_merger.lvc (output of LVTM encoding, i.e., LVTM bitstreams)

## 5.2 Running the LVTM decoder

Keep the following content in the script:

|  |
| --- |
| # decoding  python run\_lvtm.py \  --mca\_config ${root\_dir}/Boys2/Boys2\_dec.cfg \  --vvc\_vtm\_config ${vtm\_cfg} \  --vvc\_data\_config ${root\_dir}/Boys2/Boys\_vvc.cfg \  --qp 46 \  --input\_lvc\_file ${data\_dir}Boys2\_merger.lvc |

Then

bash run.sh

In this example, the following files will be produced after encoding:

* Boys2.log (decoded metadata from LVTM bitstreams)
* Boys2\_vvc.bin (reconstructed VVC bitstreams from LVTM bitstreams, as the input of the VTM decoder)
* Boys2.yuv (the yuv decoded by the VTM decoder, as the input of MCA postprocessing)
* Boys2\_dec.yuv (final output of LVTM decoding)

## 5.3 Running the RLC

Lenslet-PSNR and Multiview-PSNR are measured for lenslet video and for the multiview video rendered by Reference Lenslet content Converter (RLC-4.0, available at <https://gitlab.com/mpeg-dense-light-field/rlc>) at each test point, respectively.

The bitrate is calculated for the corresponding LVC bitstream. Then, two BD-rate [3] results, for decoded lenslet video and rendered multiview video, can be calculated using the data sheet provided at <https://content.mpeg.expert/data/CfP/LVC/DataSheet/DataSheet.xlsm>. More details can be found in the CTC[2].

# References

[1] MDS25493\_WG04\_N00715, Descriptions of core experiments on Lenslet Video Coding, ISO/IEC JTC1/SC29/WG04, Daejeon, Korea, July, 2025.

[2] MDS25492\_WG04\_N00714, Common test conditions for Lenslet Video Coding, ISO/IEC JTC1/SC29/WG04, Daejeon, Korea, July, 2025.

[3] ITU-T HSTP-VID-WPOM - Working practices using objective metrics for evaluation of video coding efficiency experiments, 2020, <http://handle.itu.int/11.1002/pub/8160e8da-en>.