INTERNATIONAL STANDARD

ISO/IEC 21794-4

First edition
2022-05

Information technology — Plenoptic image coding system (JPEG Pleno) —

Part 4:
Reference software

Reference number
ISO/IEC 21794-4:2022(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 21794 series can be found on the and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document is part of a series of standards for a system known as JPEG Pleno. The ISO/IEC 21794 series aims to provide a standard framework for representing new imaging modalities. It facilitates the capture, representation, exchange and visualization of plenoptic imaging modalities. A plenoptic image modality can be a light field, point cloud or hologram, which are sampled representations of the plenoptic function in the form of, respectively, a vector function that represents the radiance of a discretized set of light rays, a collection of points with position and attribute information, or a complex wavefront. The plenoptic function describes the radiance in time and in space obtained by positioning a pinhole camera at every viewpoint in 3D spatial coordinates, every viewing angle and every wavelength, resulting in a 7D function.

JPEG Pleno is designed primarily to facilitate the capture, representation, exchange and visualization of point cloud, light field, and holographic imaging approaches. It specifies tools for coding these approaches while providing advanced functionality at the system level such as support for data and metadata manipulation, editing, random access and interaction, protection of privacy and ownership rights as well as other security mechanisms.

This document provides reference software implementations of ISO/IEC 21794-1 and ISO/IEC 21794-2 that demonstrate the features and capabilities of JPEG Pleno for coding light field data. Its purpose is to act as a guideline for implementations and as a reference for conformance testing. As such, the implementations are conforming to ISO/IEC 21794-1, i.e. it implements the structure of a JPL file which includes the concepts of boxes. In addition, the reference software implementations also cover ISO/IEC 21794-2, i.e. light-field coding.

This document includes the source code for reference implementations of ISO/IEC 21794-1 and ISO/IEC 21794-2 as electronic attachments. They have been successfully compiled and tested on Linux®[1)] operating system at the time of writing.

Instructions for unpacking and building the software are found in Annexes A. Instructions for its use are listed in Annexes B and C.

1) Linux® is the trademark of a product supplied by Linus Torvalds. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO/IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Information technology — Plenoptic image coding system (JPEG Pleno) —

Part 4:
Reference software

1 Scope

This document provides reference implementations of ISO/IEC 21794-1 and ISO/IEC 21794-2, also known as JPEG Pleno.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 21794-1, *Information technology — Plenoptic image coding system (JPEG Pleno) — Part 1: Framework*

ISO/IEC 21794-2, *Information technology — Plenoptic image coding system (JPEG Pleno) — Part 2: Light field coding*

ISO/IEC 21794-3, *Information technology — Plenoptic image coding system (JPEG Pleno) — Part 3: Conformance testing*

3 Terms and definitions

For the purposes of this document the terms and definitions given in ISO/IEC 21794-1, in ISO/IEC 21794-2, in ISO/IEC 21794-3, and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
codestream
sequence of bytes that conforms to or is to be checked for conformance with the codestream syntax specified in ISO/IEC 21794-1 and/or ISO/IEC 21794-2

3.2
JSON
data interchange format that represents data objects using attribute–value pairs and array data types specified in ISO/IEC 21778

4 Abbreviated terms

JPLM	JPEG Pleno Model

PPM	Portable pixmap format
PSNR	Peak signal-to-noise ratio
RDO	Rate-distortion optimization
RGB	Red, Green, Blue

5 Conventions

Monospaced text indicates program input or output as necessary to either run the software, or as generated by the software on the console.

6 Reference software

The following reference software is provided:

— Reference decoder software capable of decoding codestreams that conform to ISO/IEC 21794-1 and/or ISO/IEC 21794-2.

— Sample encoder software capable of producing codestreams that conform to ISO/IEC 21794-1 and/or ISO/IEC 21794-2.

The use of the reference software is not required for making an implementation of an encoder or decoder in conformance to any of the parts of the ISO/IEC 21794 series. Requirements established in ISO/IEC 21794-1 and ISO/IEC 21794-2 take precedence over the behaviour of the reference software.

7 Examples of use

Some examples of use for the reference decoder software implementations are as follows:

— As an illustration of how to perform the decoding processes.

— As the starting basis for the implementation of a decoder that conforms to one or multiple parts of the ISO/IEC 21794 series.

— For (non-exhaustive) testing of the conformance of a codestream (or file) as specified in ISO/IEC 21794-1 or ISO/IEC 21794-2.

NOTE The lack of detection of any conformance violation by any reference software implementation is not to be considered as definitive proof that the codestream under test conforms to all constraints required for the codestream to conform to one of the ISO/IEC 21794 series.

Some examples of use for a reference encoder software are as follows:

— As an illustration of how to implement an encoding process that produces codestreams that are, depending on the settings of the software, conforming to one or multiple members of the software of ISO/IEC 21794 series.

— As a starting point for an implementation of an encoder that conforms to one or multiple members of the ISO/IEC 21794 series.

— As a means of generating codestreams conforming to one or multiple parts of the ISO/IEC 21794 series for testing purposes.

— As a means of demonstrating and evaluating examples of the quality that can be achieved by an encoding process that conforms to multiple parts of the ISO/IEC 21794 series.

No guarantee of the quality that will be achieved by an encoder is provided by its conformance to one or multiple parts of the ISO/IEC 21794 series as the conformance is defined only in terms of specific constraints imposed on the syntax of the generated codestream. In particular, while sample encoder software implementations may suffice to provide some illustrative examples of which quality can be achieved within the ISO/IEC 21794 series, they provide neither an assurance of minimum guaranteed image encoding quality nor maximum achievable image encoding quality.

Similarly, the computation resource characteristics in terms of program or data memory usage, execution speed, etc. of sample software encoder or decoder implementations should not be construed as a representative of the typical, minimal or maximal computational resource characteristics to be exhibited by implementations of some parts of the ISO/IEC 21794 series.

8 Warranty disclaimer

Regardless of any and all statements made herein or elsewhere regarding the possible uses of the reference software, the following disclaimers of warranty apply to the provided reference software implementations:

— ISO and IEC disclaim any and all warranties, whether express, implied, or statutory, including any implied warranties of merchantability or fitness for a particular purpose.

— In no event shall the contributor(s) or ISO or IEC be liable for any incidental, punitive or consequential damages of any kind whatsoever arising from the use of these programs.

— This disclaimer of warranty extends to the user of these programs and the user's customers, employees, agents, transferees, successors, and assignees.

— ISO and IEC do not represent or warrant that the software is free of infringements of any patents.

— Commercial applications of ISO/IEC International Standards, including shareware, may be subject to royalty fees to patent holders.

9 General

The reference software implementations for the ISO/IEC 21794 series are available at https://standards.iso.org/iso-iec/21794/-4/ed-1/en/.

The file "`jpeg-pleno-refsw-bbbp.zip`" contains a reference implementation for ISO/IEC 21794-1 and ISO/IEC 21794-2 that currently encodes and decodes codestreams conformant to the baseline block-based profile. Unpacking and compilation of this software is explained in Annex A, guidance on how to use this software is given in Annex B.

The file "`jpeg-pleno-refsw-bvbp.zip`" contains a reference implementation for ISO/IEC 21794-1 and ISO/IEC 21794-2 that currently encodes and decodes codestreams conformant to the baseline view-based profile. Unpacking and compilation of this software is explained in Annex A, guidance on how to use this software is given in Annex C.

10 Building the reference software

The whole build system is managed by CMake®[2]. The reference software has been compiled on a machine with Internet access, configured with the following operating system and packages:

— Linux® environment with kernel 4.15.0 or above.

— GCC 9.2.1 or above.

— CMake3.10 or above.

2) CMake® is the trademark of a product supplied by Kitware. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO/IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

— Doxygen 1.8.13 or above (optional to build binaries but required to build documentation).

— TeX Live (optional).

Internet access is required to build the reference software because CMake downloads packages to resolve library dependencies.

Annex A
(informative)

Building the reference software for ISO/IEC 21794-1 and ISO/IEC 21794-2

A.1 Building baseline block-based profile reference software

Unpacking a ZIP file is operating system specific. Under POSIX compliant operating systems, open a command line window and enter

```
~$ unzip jpeg-pleno-refsw-bbbp.zip
```

This will extract the source-code into the `jpeg-pleno-refsw-bbbp` directory.

To compile the software, please follow these steps:

```
~$ cd jpeg-pleno-refsw-bbbp
~/jpeg-pleno-refsw-bbbp$ mkdir; cd build
~/jpeg-pleno-refsw-bbbp/build$ cmake -DVISUALIZATION_TOOL=OFF ..
~/jpeg-pleno-refsw-bbbp/build$ make
```

After performing the above commands, the build can be verified via the following command:

```
~/jpeg-pleno-refsw-bbbp/build$ ctest --verbose
```

The built binaries will be stored in `~/jpeg-pleno-refsw-bbbp/bin`.

A.2 Building baseline view-based profile reference software

Unpacking a ZIP file is operating system specific. Under POSIX compliant operating systems, open a command line window and enter

```
~$ unzip jpeg-pleno-refsw-bvbp.zip
```

This will extract the source-code into the `jpeg-plm-4dpm` directory.

To compile the software, please follow these steps:

```
~$ cd jpeg-pleno-refsw-bvbp
~/jpeg-pleno-refsw-bvbp$ mkdir; cd build
~/jpeg-pleno-refsw-bvbp/build$ cmake ..
~/jpeg-pleno-refsw-bvbp/build$ make -j
```

After performing the above commands, the build can be verified via the following command:

```
~/jpeg-pleno-refsw-bvbp/build$ make test
```

The built binaries will be stored in `~/jpeg-pleno-refsw-bvbp/build/bin`.

Annex B
(informative)

Baseline block-based profile usage GUI

B.1 General

This Annex describes the usage of the software contained in the file "`jpeg-pleno-refsw-bbbp.zip`" available at https://standards.iso.org/iso-iec/21794/-4/ed-1/en/ and prepared and compiled with the instructions given in Annex A. After the building steps, the `bin` directory is created. The bin directory has the following structure:

```
|— jpl-decoder-bin
|— jpl-encoder-bin
|— tests
|    |— *_tests
|_ utils
     |— compute_lightfield_quality_metrics
     |— compute_psnr
     |— convert_pgx_to_ppm
     |— convert_ppm_to_pgx
     |— lenslet_13x13_shifter
     |_ lightfield_coordinate_shift
```

The main executable files are

`jpl-encoder-bin`, Encoder used to compress light fields into JPL files.

`jpl-decoder-bin`, Decoder used to decompress JPL files into light fields.

The executable files in `tests` directory have the suffix `_tests` and correspond to the unit tests generated with purposes of building validation.

The `utils` directory contains six useful tools. These tools are not required for the encoding and decoding processes but convenient for these tasks. The utils files are

`compute_lightfield_quality_metrics`, Compute several quality metrics for all views of an assessed light field and generate a brief report.

`compute_psnr`, Compute the PSNR for a single PPM image.

`convert_pgx_to_ppm`, Convert a single image from PGX to PPM format.

`convert_ppm_to_pgx`, Convert a single image from PPM to PGX format.

`lenslet_13x13_shifter`, Adjust dark views at corners of Lenslet datasets.

`lightfield_coordinate_shift`, Batch filename rename of light field views.

The usage instructions for these tools can be seen by calling any util app with parameter `--help`.

B.2 Using the encoder for baseline block-based profile

The encoder requires one light field input. The input light field is represented as a directory containing a set of sub-directories, one for each colour component. The name of each colour directory is the numeric index of the colour channel, starting at 0. Each of those sub-directories contains uncompressed images in PGX format. These PGX images shall be named following the “xxx_yyy.pgx” format, where “xxx” and “yyy” indicate the vertical and horizontal position of a given view in the light field. The light field shall begin at position 000_000. Furthermore, a sequence of views in a given direction shall be contiguous, meaning that the distance among views of a direction shall be 1. For instance, a 2x2 views light field with 2 colour components will have the following PGX files:

```
~/lightfield_path/0/000_000.pgx
~/lightfield_path/0/000_001.pgx
~/lightfield_path/0/001_000.pgx
~/lightfield_path/0/001_001.pgx
~/lightfield_path/1/000_000.pgx
~/lightfield_path/1/000_001.pgx
~/lightfield_path/1/001_000.pgx
~/lightfield_path/1/001_001.pgx
```

To encode a light field using the 4D transform mode the encoder binary shall be called as follows:

```
~/jpeg-pleno-refsw$ ./bin/jpl-encoder-bin [OPTIONS]
```

The options for JPLM encoder are listed in Table B.1.

Table B.1 — Options for JPLM encoder

-h,--help	Prints the help message and exits.
-c,--config	Path to a configuration file in JSON.
-ve,--verbose	Shows verbose output during execution. (Default: false)
-time,--show-runtime-statistics	Shows the total elapsed time in seconds (wall time) measured by using std::chronos. Notice that the actual execution time may be different. If UNIX, also includes user time (ms) and max memory usage (kbytes). This is the last report that will be shown after the execution. (Default: false)
-progress,--show-progress-bar	Enables the display of a progress bar showing the percentage of completion, run time and expected finishing time. (Default: false)
-i,--input	Input directory containing the plenoptic data to be compressed (according to the JPEG Pleno Part). For Part 2, light field, the input is a directory containing a set of directories (one for each colour channel). Each one of those directories contains a set of views in PGX format.
-o,--output	Output, i.e., the compressed JPEG Pleno bitstream (filename.jpl).
-xmlcat,--insert-xml-catalog	Inserts a XML box with catalogue in the generated JPL file. (Default: false)
-p,--part	The JPEG Pleno part. Mandatory. Available options are: LightField (2).

The options for JPLM encoder related to ISO/IEC 21794-2 (i.e., light field) are listed in Table B.2.

Table B.2 — Options for light field encoding

-v,--view_height	Single-view height dimension. Mandatory.
-u,--view_width	Single-view width dimension. Mandatory.
-t,--number_of_rows	Number of light-field view rows. Mandatory.
-s,--number_of_columns	Number of light-field view columns. Mandatory.
-ecs,--enum-cs	Enumerated colour space to be used in the Colour Specification Box. Currently other methods are not supported. Available values for EnumCS field: BiLevel (0), YCbCr_1 (1), YCbCr_2 (3), YCbCr_3 (4), PhotoYCC (9), CMY (11), CMYK (12), YCCK (13), CIELab (14), BiLevel_2 (15), sRGB (16), greyscale (17), sYCC (18), CIEJab (19), e_sRGB (20), ROMM_RGB (21), YPbPr_1125_60 (22), YPbPr_1250_50 (23), and e_sYCC (24). (Default: YCbCr_2)
-nc,--number-of-colour-channels	Number of colour channels. (Default: 3)

Table B.2 *(continued)*

-T,--type	Light-field codec type (mode). Available options are: transform_mode (0), and prediction_mode (1). (Default: transform_mode)

The options for JPLM encoder related to the 4D transform mode (4DTM) are listed in Table B.3.

Table B.3 — Options for the 4D transform mode (4DTM)

-TNIv,--transform_size_maximum_inter_view_vertical	Maximum 4D transform size in inter-view vertical direction. (Default: 13)
-TMIh,--transform_size_maximum_inter_view_horizontal	Maximum 4D transform size in inter-view horizontal direction. (Default: 13)
-TMiv,--transform_size_maximum_intra_view_vertical	Maximum 4D transform size in intra-view vertical direction. (Default: 64)
-TMih,--transform_size_maximum_intra_view_horizontal	Maximum 4D transform size in intra-view horizontal direction. (Default: 64)
-TmIv,--transform_size_minimum_inter_view_vertical	Minimum 4D transform size in inter-view vertical direction. (Default: 13)
-TmIh,--transform_size_minimum_inter_view_horizontal	Minimum 4D transform size in inter-view horizontal direction. (Default: 13)
-Tmiv,--transform_size_minimum_intra_view_vertical	Minimum 4D transform size in intra-view vertical direction. (Default: 4)
-Tmih,--transform_size_minimum_intra_view_horizontal	Minimum 4D transform size in intra-view horizontal direction. (Default: 4)
-B,--border_policy	Policy to treat border blocks, i.e., the blocks ranging outside the 4D limits of the light field. Available options are: padding (0), and truncate (1). (Default: truncate)
-l,--lambda	Lagrangian multiplier used in the RDO process of 4D Transform mode (floating point value). (Default: 1000.0)
-errorest,--show-error-estimate	Shows error estimates computed during RDO. Although close to the real error figures, they are only estimates (do not account for rounding errors in the inverse transform, for instance). For proper error reports, please decode the generated file and use the provided external tool. (Default: false)
-pnt,--insert-codestream-pointer-set	Inserts a codestream pointer set marker segment before the first 4D block. This enables the encoder to know where each 4D block starts. (Default: false)

B.3 Using the decoder for baseline block-based profile

The decoder requires an encoded JPL file as input. To decode such JPL file, the reference software is called as follows:

```
~/jpeg-pleno-refsw$ ./bin/jpl-decoder-bin [OPTIONS]
```

The options for the JPLM decoder are listed in Table B.4.

Table B.4 — Options for JPLM decoder

-h,--help	Print a help message and exits.
-c,--config	Path to a configuration file in JSON.
-i,--input	Input, i.e., a JPEG Pleno bitstream file (filename.jpl) to be decoded.
-o,--output	Output directory containing the decoded plenoptic data.
-ve,--verbose	Shows verbose output during execution. (Default: false)

Table B.4 *(continued)*

-time,--show-runtime-statistics	Shows the total elapsed time in seconds (wall time) measured by using std::chronos. Notice that the actual execution time may be different. If UNIX, also includes user time (ms) and max memory usage (kbytes). This is the last report that will be shown after the execution. (Default: false)
-progress,--show-progress-bar	Enables the display of a progress bar showing the percentage of completion, run time and expected finishing time. (Default: false)
-showcat,--show-xml-catalog	Shows the XML box with catalogue of the input JPL file (if present). (Default: false)

B.4 Basic encoding and decoding example

To encode a light field using the 4D Transform mode:

```
INPUT_PGX_PATH="/home/RAW/I01_Bikes_shifted_PGX/"
OUTPUT_JPL_FILE=/home/RAW/I01_Bikes.jpl

~/jpeg-pleno-refsw$ ./bin/jpl-encoder-bin --show-progress-bar \
    --part 2 --type transform_mode                               \
    --enum-cs YCbCr_2 -nc 3 -t 13 -s 13 -v 434 -u 625            \
    --border_policy truncate --lambda 10000                      \
    --transform_size_maximum_inter_view_vertical 13              \
    --transform_size_maximum_inter_view_horizontal 13            \
    --transform_size_maximum_intra_view_vertical 31              \
    --transform_size_maximum_intra_view_horizontal 31            \
    --transform_size_minimum_inter_view_vertical 13              \
    --transform_size_minimum_inter_view_horizontal 13            \
    --transform_size_minimum_intra_view_vertical 4               \
    --transform_size_minimum_intra_view_horizontal 4             \
    --input ${INPUT_PGX_PATH} --output ${OUTPUT_JPL_FILE}
```

To decode the generated JPL file:

```
OUTPUT_PGX_PATH="/home/RAW/DECODED_I01_Bikes_shifted_PGX/"

~/jpeg-pleno-refsw$ ./bin/jpl-decoder-bin --show-progress-bar \
    --input ${OUTPUT_JPL_FILE}                                   \
    --output ${OUTPUT_PGX_PATH}
```

B.5 Complete example with tools usage

Transform the `I01_Bikes` lenslet dataset from JPEG Pleno database into a valid format by taking the central 13x13 views and adjusting the darker corner views:

```
INPUT_PPM_PATH=/home/RAW/I01_Bikes/
OUTPUT_PPM_PATH=/home/RAW/I01_Bikes_shifted_PPM/
mkdir -p ${OUTPUT_PPM_PATH}

~/jpeg-pleno-refsw$ ./bin/utils/lenslet_13x13_shifter \
    --input ${INPUT_PPM_PATH}                            \
    --output ${OUTPUT_PPM_PATH}                          \
    --direction encoder
```

Convert the transformed PPM views into PGX format to be used as input in the encoder:

```
INPUT_PPM_PATH=/home/RAW/I01_Bikes_shifted_PPM/
OUTPUT_PGX_PATH=/home/RAW/I01_Bikes_shifted_PGX/
mkdir -p ${OUTPUT_PGX_PATH}

for input_view in ${INPUT_PPM_PATH}/*.ppm;
do
  ~/jpeg-pleno-refsw$ ./bin/utils/convert_ppm_to_pgx \
      --input ${input_view} --output ${OUTPUT_PGX_PATH};
done
```

Compress the light field using the PGX views as input:

```
INPUT_PGX_PATH=/home/RAW/I01_Bikes_shifted_PGX/
OUTPUT_JPL_FILE=./I01_Bikes.jpl

~/jpeg-pleno-refsw$ ./bin/jpl-encoder-bin                      \
    --show-progress-bar                                          \
    --part 2 --type transform_mode --show-runtime-statistics  \
    --enum-cs YCbCr_2 -nc 3 -t 13 -s 13 -v 434 -u 625          \
    --border_policy truncate --lambda 10000                    \
    --transform_size_maximum_inter_view_vertical 13            \
    --transform_size_maximum_inter_view_horizontal 13          \
    --transform_size_maximum_intra_view_vertical 31            \
    --transform_size_maximum_intra_view_horizontal 31          \
    --transform_size_minimum_inter_view_vertical 13            \
    --transform_size_minimum_inter_view_horizontal 13          \
    --transform_size_minimum_intra_view_vertical 4             \
    --transform_size_minimum_intra_view_horizontal 4           \
    --input ${INPUT_PGX_PATH} --output ${OUTPUT_JPL_FILE}
```

To decode the generated JPL file:

```
INPUT_JPL_FILE=./I01_Bikes.jpl
OUTPUT_DECODED_PGX_PATH=/home/RAW/DECODED_I01_Bikes_shifted_PGX/
mkdir -p ${OUTPUT_DECODED_PGX_PATH}

~/jpeg-pleno-refsw$ ./bin/jpl-decoder-bin --show-progress-bar \
    --show-runtime-statistics                                  \
    --input ${INPUT_JPL_FILE}                                  \
    --output ${OUTPUT_DECODED_PGX_PATH}
```

Get the report on quality assessment of the decoded light field:

```
PRISTINE_PGX_PATH=/home/RAW/I01_Bikes_shifted_PGX/
DECODED_PGX_PATH=/home/RAW/DECODED_I01_Bikes_shifted_PGX/

~/jpeg-pleno-refsw$ ./bin/utils/compute_lightfield_quality_metrics \
    --input_baseline ${PRISTINE_PGX_PATH}                          \
    --input_test ${DECODED_PGX_PATH}                               \
    -t 13 -s 13 --show-report-all
```

Convert decoded PGX views back to original PPM format:

```
INPUT_DECODED_PGX_PATH=/home/RAW/DECODED_I01_Bikes_shifted_PGX/
OUTPUT_DECODED_PPM_PATH=/home/RAW/DECODED_I01_Bikes_shifted_PPM/
mkdir -p ${OUTPUT_DECODED_PPM_PATH}

for i in ${INPUT_DECODED_PGX_PATH}/0/*.pgx;
do
    view_filename=$(basename ${i});
    view="${view_filename%.pgx}";
    ~/jpeg-pleno-refsw$ ./bin/utils/convert_pgx_to_ppm  \
       --input ${INPUT_DECODED_PGX_PATH}                \
       --output ${OUTPUT_DECODED_PPM_PATH}/${view}.ppm
done
```

Compute PSNR of each decoded PPM format in RGB colour space:

```
PRISTINE_SHIFTED_PPM_PATH=/home/RAW/I01_Bikes_shifted_PPM/
DECODED_SHIFTED_PPM_PATH=/home/RAW/DECODED_I01_Bikes_shifted_PPM/

for i in ${PRISTINE_SHIFTED_PPM_PATH}/*.ppm;
do
    view=$(basename ${i});
    ~/jpeg-pleno-refsw$ ./bin/utils/compute_psnr    \
      --input ${PRISTINE_SHIFTED_PPM_PATH}/${view}  \
      --output ${DECODED_SHIFTED_PPM_PATH}/${view}  \
      --color_space RGB
done
```

Annex C
(informative)

Baseline view-based profile usage guide

C.1 General

This Annex describes the usage of the software contained in the file "jpeg-pleno-refsw-bvbp.zip" (available at https://standards.iso.org/iso-iec/21794/-4/ed-1/en/) and prepared and compiled with the instructions given in Annex A.

The main executable files are:

`jpl-bvbp-encoder`, Encoder used to compress light fields into JPL files.

`jpl-bvbp-decoder`, Decoder used to decompress JPL files into light fields.

C.2 Using the encoder for baseline view-based profile

The encoder requires one light field input. The input light field is represented as a directory containing a set of uncompressed images in ppm format. These ppm images shall be named following the “xxx_yyy.ppm” format, where “xxx” and “yyy” indicate the vertical and horizontal position of a given view in the light field. The 4D Prediction mode tools for sample-based warping, merging and prediction require disparity information of the input light field. The input disparity information is represented as one uncompressed image in pgm format (e.g. xxx_yyy.pgm), where “xxx” and “yyy” indicate the vertical and horizontal position of the view corresponding to the disparity information.

To encode a light field using the 4D prediction mode the encoder binary shall be called as follows:

```
JPL Reference Software Baseline View Based Profile (4DPM) Encoder
Usage: ./jpl-bvbp-encoder [OPTIONS]
```

The options for the 4DPM encoder are listed in Table C.1.

Table C.1 — Options for the 4DPM encoder

Parameter	Description
-h, --help	Print this help message and exit
-i, --input	Input directory containing a set of uncompressed light-field images (xxx_yyy.ppm).
-o, --output	Output directory containing temporary light-field data and the compressed bitstream.
-c,--config	TEXT REQUIRED Path to JPEG Pleno Baseline View Based Profile config file
-s,--codestream	Output codestream (default output.jpl)
-x, -disablecommoncodestream	Enable(default)/0=Disable removal of the JPEG 2000 main header and move it in the common codestream element)
-r, --reversible	Enable reversible transform (lossless)

JPEG Pleno Baseline View Based Profile config file is a JSON file with the following structure:

```
{
    HEADER
    "views": [ { VIEW1 },
            { VIEW2 },
            ...    ]
}
```

The JSON configuration file HEADER parameters are given in Table C.2:

Table C.2 — JSON configuration file HEADER parameters

Parameter	Value	Description
type	{"4DPM"}	JSON configuration file identifier
colorspace	{"RGB", "YCbCr"}	defines the colour space used by the encoder (and decoder) in internal processing. Input and final output are in RGB xxx_yyy.ppm with values in range of [0,1023]. The RS supports 10-bit images.
sai_rows	[1,T]	number or rows in the sub aperture image (SAI) reference plane grid
sai_columns	[1,S]	number or columns in the SAI reference plane grid
image_height	[1,2^32-1]	SAI height, all the SAI have the same height
image_width	[1,2^32-1]	SAI width, all the SAI have the same width
number_of_views	[1,2^32-1]	number of views to be encoded using the corresponding parameters specified in the following view array. The number_of_views shall be equal to the number of input ppm files.

JSON Configuration file VIEWx parameters are given in Table C.3:

Table C.3 — JSON Configuration file VIEWx parameters

Parameter	Value	Description
id	[0,number_of_views-1]	index of the current SAI in coding order. First SAI has id zero (id=0), and belongs to level one (level=1)
level	[1, H_max],	hierarchical level of the SAI, where H_max is the highest hierarchical level
row_index	[yyy]	SAI row index (yyy in xxx_yyy.ppm input file)
column_index	[xxx]	SAI column index (xxx in xxx_yyy.ppm input file)
horizontal_camera_center_position	[float]	value for horizontal camera center position
vertical_camera_center_position	[float]	value for vertical camera center position
rate_texture	[float], bpp,	rate in bpp for encoding the texture prediction residual, the bit budget for encoding the texture residual for xxx_yyy.ppm is (rate_texture width height)

Table C.3 *(continued)*

Parameter	Value	Description
number_of_texture_references	[0,N]	the number of SAIs used in texture view synthesis,, where N is the number of views with hierarchical level smaller than “level”
texture_reference_indices	[id_1, id_2, ...],	the id’s of the SAIs used in texture synthesis for the current SAIs, vector of id’swith each corresponding to SAIs on hierarchical levels smaller than current SAI level
view_merging_mode	{0,1,2},	used when view_merging_mode=1. During view merging, larger values give more influence to faraway camera position. A good default value is for example 10. The formula for weighting a view is, exp(-distance/(2*fw^2)), where fw equals fixed_merging_weight_parameter, and distance is the Euclidean distance between the two views computed from
fixed_merging_weight_parameter	[uint]	used when view_merging_mode=1. During view merging, larger values give more influence to faraway camera position. A good default value is for example 10. The formula for weighting a view is, exp(-distance/(2*fw^2)), where fw equals fixed_merging_weight_parameter, and distance is the Euclidean distance between the two views computed from horizontal_camera_center_position and vertical_camera_center_position
sparse_filter_neighborhood_size	{0,...,min(width,height)}	size of the sparse filtering template. The template is rectangular area centered on each pixel location and has dimensions (2*M+1)*(2*M+1), where M=sparse_filter_neighborhood_size
sparse_filter_order	{0,..., (2*M+1)*(2*M+1)+1},	the order of the sparse filter
cweight_search	{0,1}	enable/disable search for the best weighting parameter for JPEG 2000 encoding applied to channel zero when colour space is YCbCr
rate_inverse_depth	[float], bpp	rate in bpp for encoding the inverse depth, the bit budget for encoding the inverse depth in xxx_yyy.pgm is (rate_texture width height)
number_of_inverse_depth_references	[0,N]	the number of SAIs used in inverse depth synthesis, where N is the number of views with hierarchical level smaller than *level*
inverse_depth_reference_indices,	[id_1, id_2, ...]	the id’s of the SAIs used in inverse depth synthesis for the current SAIs, vector of id’s with each corresponding to SAIs on hierarchical levels smaller than current SAI level

Table C.3 *(continued)*

Parameter	Value	Description
minimum_inverse_depth	[uint]	inverse depth level shift, the inverse depth is a real valued array, but is stored as a quantized integer valued xxx_yyy.pgm file. Level shifting and division by 2^14 (dequantization) is used to obtain the original real valued array. In level shifting the value "minimum_inverse_depth" is subtracted from xxx_yyy.pgm prior to dequantization

The reference software contains a set of JSON configuration samples. A short description of each one is shown in Table C.4.

Table C.4 — JSON Configuration sample descriptions

Sample name	Description of the configuration
config/lenslet_13_13_allIntra	Lossless compression of all the LF
config/lenslet_13_13_6layers	Lossy compression of 13x13 SAI, on a grid 15x15, with 1 disparity for the central SAI, and six hierachical levels.

C.3 Using the decoder for baseline view-based profile

The decoder requires one encoded JPL file input. The input JPL is formatted according to ISO 21794-1 and 2 and generated by the encoder as described in A.2.

```
JPL Reference Software Baseline View Based Profile (4DPM) Decoder

Usage: ./jpl-bvbp-decoder [OPTIONS]
```

The options for the 4DPM decoder are listed in Table C.5.

Table C.5 — Options for the 4DPM decoder

Parameter	Description
-h,--help	Print this help message and exit
-i,--input	Input JPEG Pleno Light Field codestream
-o,--output	Output directory the decoded light-field in format xxx_yyy.ppm

C.4 Examples

Encoding a light field using the 4D Prediction mode displays the number of levels under coding and and for each view encoded at a given level a dot is shown (as coding progress signal). Finally, the input path and config and the output codestream filename are displayed.

```
$ ./jpl-bvbp-encoder -i ./lightfieldpath -o ./outputpath -c ./config/file.json -s
lightfield.jpl

[HLEVEL=1]
.
[HLEVEL=2]
........
[HLEVEL=3]
............
[HLEVEL=4]
........................
[HLEVEL=5]
```



```
...................................
[HLEVEL=6]
......................................................................................
Input:
- light field: /Users/jpeg/lightfieldpath
- config file: /Users/jpeg/config/file.json
Ouput:
- JPEG Pleno Light Field codestream
  /Users/jpeg/encoded.jpl
```

Decoding a 4D prediction mode JPEG Pleno codestream:

```
$ ./jpl-bvbp-decoder -input ./lightfield.jpl -output ./outputpathdec

H levels: 6
........................................................................................
...........
..................................................................
Input - JPEG Pleno Light Field codestream:
  /Users/jpeg/encoded.jpl
Output - light field:
  /Users/jpeg/outputpathdec
```

More examples can be found in the example folders available at https://standards.iso.org/iso-iec/21794/-4/ed-1/en/.

ICS 35.040.30

Price based on 15 pages