JPEG Pleno light field coding technologies

Peter Schelkens[a,b], Pekka Astola[c], Eduardo A. B. da Silva[d], Carla Pagliari[e], Cristian Perra[f], Ioan Tabus[c], and Osamu Watanabe[g,a]

[a]Vrije Universiteit Brussels, Belgium
[b]imec, Leuven, Belgium
[c]Tampere University (TAU), Finland
[d]PEE/COPPE/DEL/POLI/UFRJ, Universidade Federal do Rio de Janeiro, Brazil
[e]PGEE/PGED/IME, Instituto Militar de Engenharia, Rio de Janeiro, Brazil
[f]DIEE, UdR CNIT, University of Cagliari, Italy
[g]Takushoku University, Tokyo, Japan

ABSTRACT

JPEG Pleno provides a standard framework to facilitate the capture, representation, and exchange of light field, point cloud and holographic imaging modalities. JPEG Pleno Part 2 addresses coding of light field data. Two coding modes are supported for this modality. The first mode exploits the redundancy in this 4D data by utilizing a 4D transform technique, the second mode is based on 4D prediction. Both techniques are outlined in this paper as well as the file format that encapsulates the resulting codestreams.

Keywords: JPEG Pleno, light field, coding technologies, standards

1. INTRODUCTION

Point cloud, light field, and holography are novel image modalities finding applications in several domains such as medical applications, industrial application, immersive communication, gaming, and more. These image modalities are different sampled representations of the plenoptic function which describes for every point in 3D space the amount of light that is radiated in every direction for every wavelength and every time instance. A point cloud representation consists of a collection of points with position and attribute information. A light field representation is a vector function representing the radiance of a discretized set of light rays. A hologram represents the plenoptic function as a complex wavefront.

Conversion from one image modality to another is often required. For example, a rendering application can operate on a different representation (e.g. holography) than the acquisition domain (e.g. point cloud) or processing domain (e.g. light field). A common framework for facilitating the acquisition, conversion, processing, and rendering of these image modalities is required for fostering the development of novel application and for providing interoperability between different systems and technologies.

In 2015, the JPEG committee, ISO/IEC JTC1/SC29/WG1, started a new project, named JPEG Pleno,[1] for the investigation of possible coding tools for these novel image modalities. In 2016, a Call for Proposals for light field coding was issued. In 2017, the submitted proposals have been evaluated and analyzed through several core experiments aiming at identifying the relevant component and tools.

The JPEG Pleno specification (ISO/IEC 21794) is currently composed of the four parts shown in Fig. 1.

Further author information: (Send correspondence to P. Schelkens)
P. Schelkens: E-mail: peter.schelkens@vub.be
P. Astola: E-mail: pekka.astola@tuni.fi
E.A.B. da Silva: E-mail: eduardo@smt.ufrj.br
C. Pagliari: E-mail: carla@ime.eb.br
C. Perra: E-mail: cperra@ieee.org
I. Tabus: E-mail: ioan.tabus@tuni.fi
O. Watanabe: E-mail:owatanab@es.takushoku-u.ac.jp

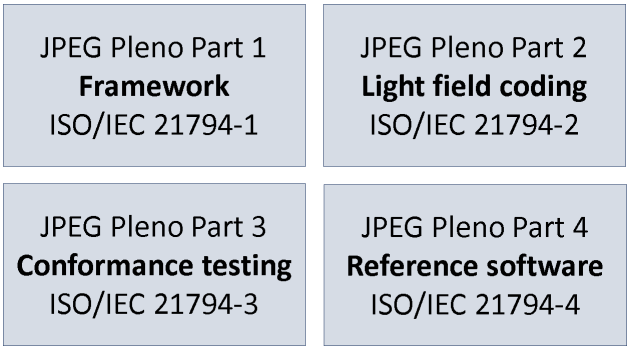

Figure 1. JPEG Pleno specification.

Part 1, "JPEG Pleno: Framework", provides a general description of the framework and specifies the generic file format. Part 2, "JPEG Pleno: Light Field Coding" specifies the file format and codestream syntax for coded light fields, the normative output of the light field encoders together with informative documentation for the encoding and decoding procedures. Part 3, "JPEG Pleno: Conformance Testing" specifies the conformance testing necessary for assessing whether third-party implementations operate within compliance bounds. Part 4, "JPEG Pleno: Reference Software" is the software implementations of the JPEG Pleno framework to be used as reference for compliance testing.

Since the standardization process started, several exploration and experimental studies have been conducted by the JPEG experts compare different coding tools applied to light field image coding, to assess the performance in terms of objective and subjective image quality evaluation,[2] and to evaluate the performance of the JPEG Pleno light field coding tools.[3]

The paper is organized as follows. Section 2 summarizes the generic file format of the JPEG Pleno framework. The JPEG Pleno light field coding architecture, and the two supported coding modalities are presented in Section 3. Section 4 draws the conclusions.

2. JPEG PLENO GENERIC FILE FORMAT

2.1 Overview

JPEG Pleno Part 1 defines a file format, which provides a foundation for storing application specific data (metadata) in association with a JPEG Pleno codestream, such as information required to process or render the content. The name of this file format is JPL. The JPL format is based on the box-based file format issued in the context of JPEG 2000,[4,5] which is on its turn based on the ISO Base Media File Format (ISO/IEC 14496-12 MPEG-4 Part 12).

As shown in Fig. 2, all information contained within the JPL file is encapsulated in boxes. Note that boxes with dashed borders are optional in conforming JPL files. There are several types of boxes; the definition of each specific box type defines the kinds of information that may be found within a box of that type. Some boxes will be defined to contain other boxes. At the highest level, the box-based file format is a simple binary container. One reason for adopting the box-based structure as the JPEG Pleno file format is that it provides a mechanism to encapsulate both the compressed codestream(s) and any metadata required to describe the encapsulated data. The binary container also provides applications the possibility to gain efficiently access to the data embedded in the file. The other reason is that the box-based file format is built upon a strong foundation with well-defined means for extension.

In the JPL file format, the JPEG Pleno Signature box defines the file is a JPEG Pleno container. Next, the file type box provides information on file type, version and compatibility information. The JPEG Pleno Thumbnail box allows for signalling a thumbnail image providing a snapshot of the plenoptic content contained, without needing to decode the contained plenoptic content. Note that this snapshot image can be quite advanced and representing for example an hyperspectral image of an hyperspectral light field. The JPEG Pleno Light Field

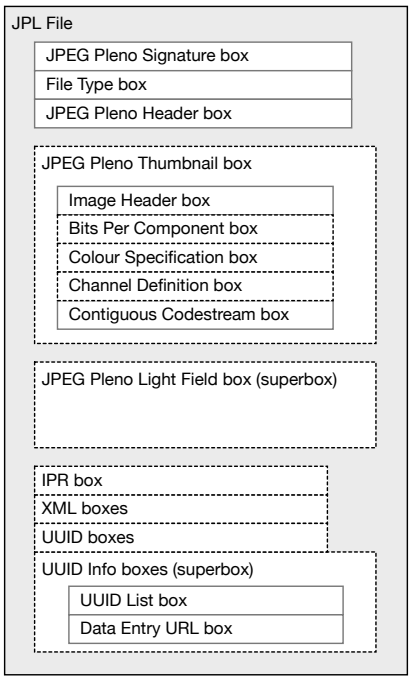

Figure 2. Conceptual structure of a JPL file: Boxes with dashed borders are optional.

superbox is being discussed in the next section. At the time of this writing, the coding technologies for point cloud and holography are not defined in the JPEG Pleno specifications. However, when those technologies become ready to be included as a part of the JPEG Pleno, the JPL format can be extended to truly meet the needs of the applications for point cloud and/or holography.

Thereafter, additional boxes providing metadata can be provided: an IPR box (providing intellectual property rights information related to the carried content), an XML box, UUID (universally unique identifier) boxes (containing after an arbitrary 16 byte identifier e.g. Exif or IPTC payload) and the associated UUID infoboxes.

The standard also foresees a particular and optional XML box containing the JPEG Pleno file catalog. This catalog provides information about where in the JPL file the contained superboxes can be found without causing the need to parse the file to find the requested superbox. This mechanism has been included to counteract the potentially huge file sizes associated with plenoptic modalities.

Finally, also a global and local reference grid has been defined. This grid allows to position multiple plenoptic modalities in a global coordinate system by signalling for every contained modality its exact position and rotation. However, when processing a particular superbox, all encoded content is supposed to be positioned with the local reference grid system associated to this superbox. This mechanisms allows for example to support the segmentation of large point clouds in smaller units, or to support the conversion between different modalities, e.g. positioning a hologram plane at the right projection distance and angle from the point cloud from which it was generated.

In the next section, we will focus on the content of the JPEG Pleno Light Field Superbox.

2.2 JPEG Pleno Light Field superbox

Figure 4 shows the hierarchical organization of the JPEG Pleno Light Field box contained by a JPL file. The file can contain multiple instantiations of a particular box type.

The JPEG Pleno Light Field superbox contains also a JPEG Pleno Thumbnail box to optionally signal a snapshot of the contained light field (e.g. a picture from one particular viewing position). In addition it contains a JPEG Pleno Light Field Header box signalling parameterization information about the light field such as size and color parameters, but also on the coding mode that is activated: 4D prediction or 4D transform mode.

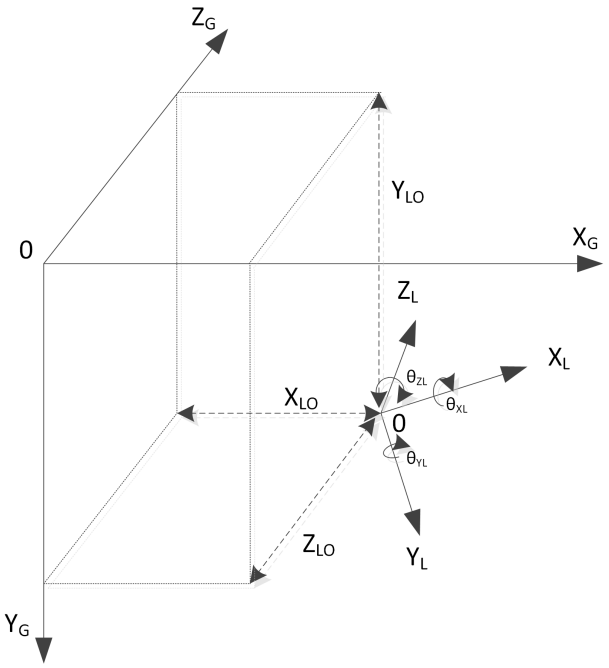

Figure 3. Global and local reference grid system for JPEG Pleno modalities.[6]

Optionally, also a Camera parameters box can be signalled providing information on the positioning of the local reference grid in the global reference grid, its size, and calibration information about the light field. The latter information is particularly important in the 4D prediction mode, where intermediate views are predicted from reference frames and and associated inverse depth maps. The exact spatial positioning of these different views is of utmost importance for optimal prediction performance. This box is optional.

In case of the 4D transform mode this box is followed by a Contiguous Codestream box that holds the coded light field utilizing this mode. In the 4D prediction mode, the following core elements are defined:

- a JPEG Pleno Light Field Reference View box containing the compressed reference views of the light field;
- a JPEG Pleno Light Field Inverse Depth View box signalling disparity information for all or a subset of subaperture views;
- a JPEG Pleno Light Field Intermediate View box containing prediction parameters and possible compressed residual signals for subaperture views not encoded as reference views.

Thereafter, several types of metadata boxes can be signalled, containing for example also vendor specific information.

3. JPEG PLENO LIGHT FIELD CODING

3.1 General Architecture

The general architecture of the light field encoder is shown in Fig. 5. The inputs to the light field encoder are the light field data to be compressed, the corresponding camera parameters, and depth maps of the light field. Two coding modes are supported. The 4D prediction mode exploits prediction as main tool for data compression while the 4D transform mode is based on 4-dimensional Discrete Cosine Transform (4D-DCT) and block partitioning. This section provide a detailed description of the two coding modes while the encapsulation of the codestream in the file format has been described in the previous section.

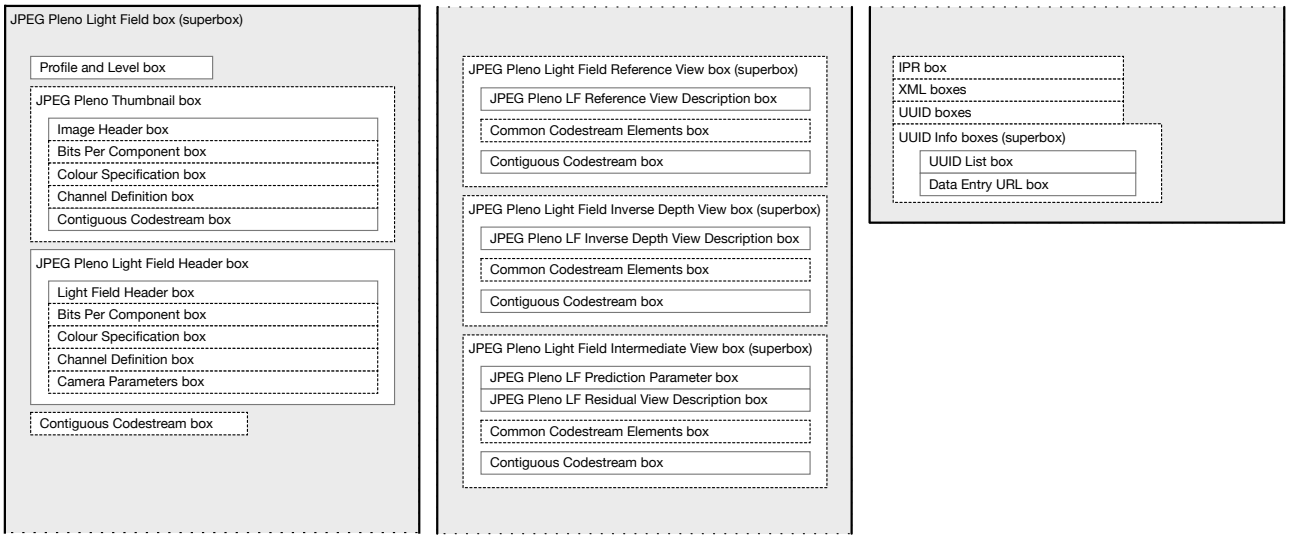

Figure 4. Hierarchical organization of a JPEG Pleno Light Field superbox: Boxes with dashed borders are optional.

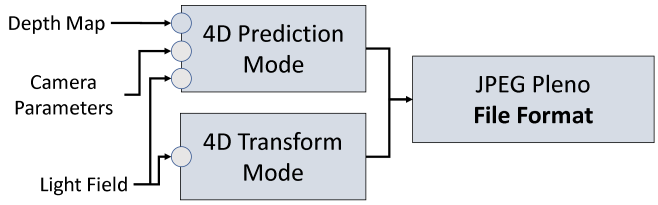

Figure 5. General architecture of JPEG Pleno light field encoder.

3.2 4D Prediction mode

The 4D prediction mode[6] begins by dividing the views into subsets with each subset corresponding to a particular hierarchical level. Each view is predicted by depth based warping of the pixels from neighboring views belonging to a lower hierarchical level. The views at the lowest hierarchical level are known as reference views and are encoded using a normative 2D codec labeled as the external codec for texture data. Each view at the lowest hierarchical level is associated with an inverse depth view, which together with the camera calibration data allows for efficient prediction of the intermediate views at higher hierarchical levels using depth based view warping. The inverse depth views are encoded with the external codec selected for the inverse depth views. The default option is to encode both texture and inverse depth of the reference views using JPEG 2000 but other codecs from the JPEG family are also supported such as JPEG, JPEG LS, JPEG XR, or JPEG XS. The Contiguous Codestream boxes in the Light Field Reference View box and in the Light Field Inverse Depth View box contain the texture and inverse depth codestreams respectively, see Figure 6. The Common Codestream Elements boxes are used to store redundant data, such as dimensions, bit depth and other parameters for each type of externally coded image data.

A summarized version of the 4D view prediction mode is shown in Figure 7. The depth-based warping and the optimal linear prediction based merging of multiple warped reference views provide the 4D prediction tools of the JPL format. After the view warping and merging stage a final adjustment of the texture is done using a supplemental sparse filter where the support is selected using a greedy sparse algorithm. Finally, the view prediction residual is decoded using an external 2D codec such as JPEG 2000. Similar to the texture, the inverse depth is synthesized at each intermediate view location using depth based warping. The synthesized inverse depth views are later used to obtain the pixel displacements when predicting the texture of the neighboring views.

While the views on the lowest hierarchical level are labeled as reference views, we also use the term reference

view when discussing any view used in predicting an intermediate view. This follows from the fact that during the hierarchical encoding and decoding, an intermediate view may later serve as a reference view to its neighboring intermediate views.

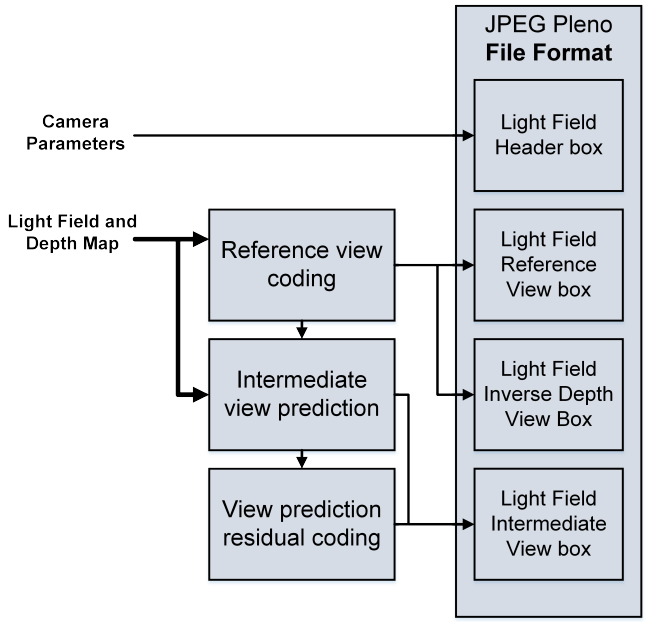

Figure 6. 4D Prediction mode encoder.

3.2.1 Hierarchical coding of sub-aperture views

The 4D prediction mode exploits the inter-view redundancy of the light field via depth based warping.[7] To reduce the number of occluded pixels the reference views should be selected so that they surround an intermediate view both horizontally and vertically in the view array. This requirement implies a certain order for the efficient coding of the views and this ordering can be realized by dividing the views into disjoint subsets. In the view hierarchy, each subset of views can be efficiently predicted from the lower subsets with the exception of the lowest subset. Often the lowest subset consists of the corners plus center configuration and the views in the higher hierarchical levels are positioned in between the already coded subsets. Since each hierarchical level contributes a set of reference views for higher hierarchical levels, the views on the higher hierarchical levels are more efficiently coded compared to the lower hierarchical levels. For each view in each hierarchical level, the full cycle of view prediction and view prediction residual coding is performed before moving to a higher hierarchical level.

Differently than the hierarchical coding of the texture, the inverse depth views are always predicted only from the inverse depth views corresponding to the lowest hierarchical level. Since no prediction residual is used for inverse depth views this restriction ensures that significant accumulation of prediction errors does not occur.

The JPL format divides the view hierarchy information into both the Light Field Reference View box and the Light Field Intermediate View box. The Light Field Reference View Description box contains a binary array with the initial division of the views between the lowest hierarchical level and the rest. The Light Field Intermediate box assigns for the rest of the views the corresponding view hierarchy levels. The reference view configuration for each intermediate view is provided in the Light Field Intermediate box as a part of the contiguous codestream of view prediction parameters.

3.2.2 View warping and merging

The inverse depth views together with the camera calibration data in the Additional Information box of the Light Field Header box can be used to obtain the pixel correspondences (i.e., horizontal and vertical displacements) between all of the views in the light field. The view warping algorithm applies the horizontal and vertical

displacements to the pixels of the reference views and thus obtains a warped version of the reference views, which serve as predictions of the intermediate view. The warping from many reference views to a same intermediate view location will result in multiple estimates at many pixels, and a mechanism for providing a single, optimal value, based on several available estimates is needed. This prediction is also known as view merging, which in the JPL format is realized using a segmentation based algorithm. The pixels at the intermediate view are labeled based on their occlusion class. Each occlusion class belongs to a particular combination of non-occluded reference views. The view warping stage merges the texture of the warped reference views using optimal linear prediction designed independently for each occlusion class. The optimal merging coefficients are provided in the Light Field Intermediate box as a part of the contiguous codestream of view prediction parameters.

For encoding at very low bit rates, the JPL format provides two low rate view merging modes. In the first one, the view merging weights are obtained from the Euclidean distance between the reference views and the intermediate view. This results in a very light overhead: instead of the full set of optimal merging coefficients only one scaling parameter is needed as a part of the contiguous codestream of view prediction parameters. Additionally, a parameter free median filter based view merging mode is provided as a fail-safe option for extremely low rates and other cases when for example the camera configuration parameters from the Light Field Header box are not available. The median filter based view merging mode is also the only view merging mode used in the synthesis of the inverse depth views.

After view merging some of the pixel locations may remain undefined due to occlusions. However, by selecting a suitable view hierarchy the number and density of undefined pixels remain low. Therefore, the values at the missing pixel locations are filled by an iterative inpainting approach where the value at a missing pixel location is obtained as the median of its neighbouring defined pixel values.

The final adjustment of the predicted intermediate view is performed using a 2D filter with its support selected using a greedy sparse algorithm. This optional filtering step is used to lower the Mean Squared Error (MSE) of the prediction errors prior to view residual coding. The sparse filter parameters are provided in the contiguous codestream of view prediction parameters.

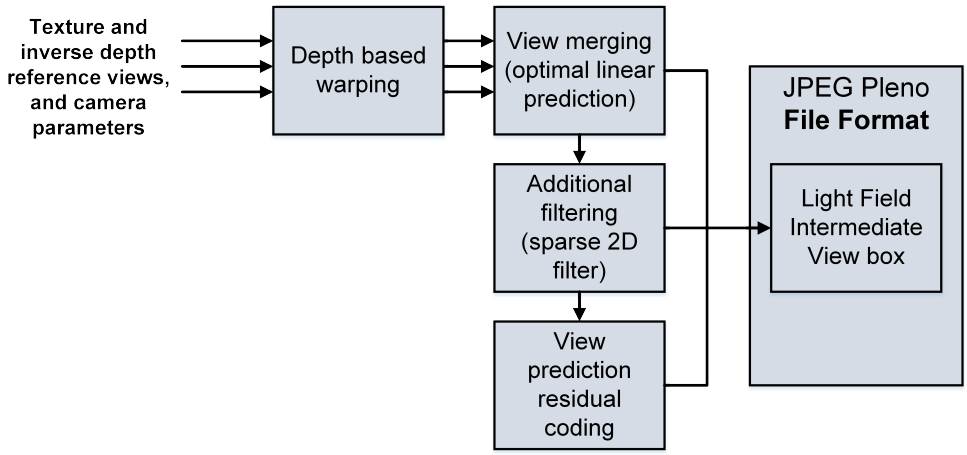

Figure 7. The different stages of the intermediate view prediction in the 4D texture prediction. The view merging coefficients, the sparse filter parameters, and the codestream for the view prediction residual are all stored in the Light Field Intermediate View box.

3.2.3 View prediction residual

The view prediction stage is optionally followed by the encoding of the view prediction residual. As explained in Section 3.2.1 the view prediction performance increases towards the higher hierarchical levels and for a fixed quality the required code length for the view prediction residual decreases. For low and medium bit rates, the view prediction residual is not necessarily encoded at the higher hierarchical levels. However, for very high quality coding at higher rates the view prediction residual is encoded on all hierarchical levels. Similarly to the

encoding of the texture and inverse depth of the reference views, the encoding of the view prediction residual is done using one of the supported codecs from the JPEG family. The codestreams for the encoded view prediction residuals are stored in the Contiguous Codestream box of the Light Field Intermediate View box.

3.3 4D Transform mode

In the 4D transform mode,[6] the light field is encoded with a four-step process (Figure 8). First, the 4D light field data is divided into fixed-sized 4D blocks that are independently encoded according to a predefined and fixed scanning order. If any of the light field dimensions are not multiple of such fixed-sized 4D blocks, the sizes of the 4D blocks at the light field boundaries have to be truncated to fit in the light field dimensions. The initial blocks can be further partitioned into a set of non-overlapping 4D sub-blocks, where the optimal partitioning parameters are derived based on a rate-distortion (R-D) criterion. Each sub-block is independently transformed by a variable block-size 4D DCT transform. Subsequently, the transformed blocks are quantized and entropy coded using hexadeca-tree bit plane decomposition and adaptive arithmetic encoding, producing a compressed representation of the light field. This coding procedure is applied to each colour component independently (e.g., Y, C_b and C_r).

In brief, the encoder adheres to the classical three-steps paradigm: transformation, quantization and entropy coding (although one can argue that the hexadeca-tree bit plane block performs both quantization and entropy coding).

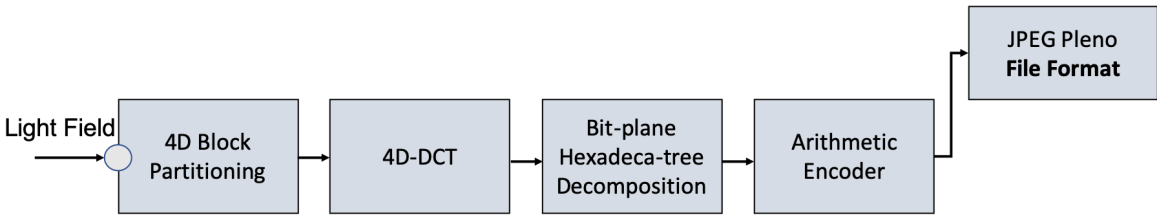

Figure 8. 4D Transform mode encoder.

3.3.1 4D Block Partitioning

The encoder depicted in Figure 8 exploits the (significant) spatial-view correlation of the 4D structure employing a hierarchical decomposition-based scheme represented by the 4D Block Partitioning module. The objective is to find the binary splits/segmentations that maximize the bit rate savings according to the Lagrangian cost J, i.e. minimizing the R-D cost: $J = D + \lambda R$. Each binary split is controlled by a ternary flag that signals a quad-tree split in the intra-view dimensions, a quad-tree split in the inter-view dimensions, or no split.

These binary splits can be represented by a segmentation tree ($\mathcal{T}$). R-D optimization of the block partitioning is performed evaluating the Lagrangian cost for every segmentation (splitting) decision, given by: $J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T})$. For a given value of the Lagrangian multiplier (λ), $D(\mathcal{T})$ is the resulting distortion when using the tree $\mathcal{T}$ and $R(\mathcal{T})$ is the associated rate.

3.3.2 4D transform

The DCT is well-known to concentrate the energy of redundant signals in its lower frequency coefficients.[8] The 4D-DCT is applied to each light field spatial dimension (separable transform). It thus can take advantage of the 4D redundancy to generate coefficients whose energy is well concentrated in the 4D frequency space. The 4D transform is applied to the 4D blocks resulting from the 4D block partitioning step, and it thus a variable-size 4D-DCT.

Since the DCT generates coefficients that tend to be concentrated in the lower 4D frequencies, after quantization most higher frequency coefficients tend to be zero. Thus an efficient encoding method would be one that can efficiently represent these zeros. When the 2D-DCT is employed, an effective way is to perform zig-zag scanning followed by run-length encoding in order to skip the zeros, thus encoding the non-zero coefficient location information using as less symbols as possible.[8] But there are alternative ways to encode the positions of the non-zero coefficients, such as the use of quad-trees.[9]

3.3.3 Bit plane Hexadeca-tree Decomposition

When using quad-tree structures each node of a quad-tree divides a 2-D region into four sub-regions. If a sub-region contains only zero coefficients or a single non-zero coefficient, it is not further subdivided and a symbol '0' is encoded. Otherwise, the subregion is further subdivided into four subregions and a symbol '1' is encoded. This subdivision process proceeds recursively until no further subdivision can be performed. At the end of this process, the positions of all the non-zero coefficients are encoded by this '0's and '1's bitstream. The quad-tree is the data structure that represents the whole subdivision process. A '0' symbol in a quadtree is thus a way to indicate that all the coefficients inside a region are zero; therefore, if this region is large, this '0' symbol is very efficient in signaling that all those coefficients are zero. Due to this property, quad-trees have been successfully used in image codecs, the one described in[10] being a good example.

As the 4D Transform mode codec uses a 4D-DCT, the equivalent to a quad-tree would be to perform subdivisions of 4D regions of coefficients into 16 4D subregions. Thus, a '0' indicating no further subdivision of a 4D region is an extremely efficient way to represent many zeros using just one symbol (an $N \times N \times N \times N$ region contains N^4 zeros). One can call such a data structure an hexadeca-tree, where the "hexadeca" prefix stands for division of a 4D region into 16 4D subregions.

In the 4D-DCT mode of the JPEG-Pleno codec, after the 4D-DCT is performed, the set of transform coefficients is sliced into 4D bit planes. The first bit plane (bit plane 0) contains the least significant bits (LSB) of the 4D-DCT coefficients, and the last bit plane $P-1$ contains their most significant bits (MSB). The number P of bit planes depends on the bit depth of the image and on the scaling used for the DCTs of each dimension. The number of bit planes in a block is determined by the largest absolute value of the 4D-DCT coefficients of the block.

A coefficient is considered non-significant on a bit plane if its bits belonging to the bit planes that are more significant than the current one are all zero. Otherwise, the coefficient is considered to be significant. The hexadeca-tree is used to group the non-significant coefficients and thus localize the significant coefficients.

As rate-distortion optimization improves coding efficiency, the quantization and entropy encoding rely on the R-D optimized hexadeca-tree structure. This tree is uniquely represented by a series of ternary flags, built by recursively subdividing a 4D block until all significant coefficients correspond to a 4D sub-block of $1 \times 1 \times 1 \times 1$ size. The ternary flags signal that either a block of 4D-DCT coefficients containing a significant coefficient at the current bit plane is split into 16 blocks in the four t, s, v, u (light field) dimensions, or a block of 4D-DCT coefficients not containing any significant coefficient at the current bit plane is not split and encoded as all-zeros, or a block of 4D-DCT coefficients containing a significant coefficient at the current bit plane is discarded (i.e, not split and encoded with all-zeros).

The 4D bit planes are scanned from the most significant to the least significant bits, with the least significant 4D bit plane being determined by the desired quantization level. Both the hexadeca-tree bits and the bits from the coefficients are encoded using an adaptive arithmetic coder, together with the 4D block partitioning flags (see Section 3.3.1).

The rate and the distortion achieved depend heavily on the choice of the segmentation tree as well as the data itself, and those should be matched. In other words, both the 4D block partitioning tree and the hexadeca-tree are determined by Lagrangian optimization using the same global Lagrange multiplier λ. Indeed, the same Lagrange multiplier should be used for all color channels.

3.3.4 Arithmetic Encoder

Since the statistics of the 4D-DCT coefficient bits, sign bits and partition flags are dependent on the coding context, the bitstream generated is encoded using a context-based adaptive binary arithmetic coder.[11] The 4D block partition ternary flags are first encoded with a 2 bit binary code and the resulting bits are arithmetic encoded using a non-adaptive context. The hexadeca-tree ternary partition flags are also first encoded with a 2 bit binary code that is further arithmetic encoded using 64 different adaptive contexts. The DCT coefficient sign bit is encoded using a single fixed context and the 4D-DCT coefficient bits are encoded using 32 adaptive contexts. The choice of the contexts is dependent on the current bit plane and the binary tree level. All the adaptive contexts are reset for each new fixed-size 4D block.

3.3.5 Remarks

It is important to observe the 4D Transform mode encoder tries to exploit the light field's 4D redundancy as a whole. One of its advantages is that it does not need depth data, which is not always readily available with the accuracy needed for the success of the light field encoding methods based on view synthesis. However, this comes with a disadvantage: the 4D transform codec will have good R-D performance if the light field has sufficient inter-view redundancy, which is the case for very high angular density light fields such the ones from the Lenslet datasets. The performance of the 4D transform mode for Lenslets[12] is indeed very good, tending to be better than the one of the 4D Prediction mode. However, for more sparse light fields, the 4D transform mode has a significant loss in performance. For these cases, the best solution is the one provided by the 4D Prediction mode described in Section 3.2.

Another important feature of the 4D transform mode is that, prior to encoding, the light field is divided into fixed-size 4D blocks that are fed to the encoder. Since these fixed-size blocks are encoded independently of each other, the 4D transform mode provides straightforward random access to these fixed-size 4D blocks, which can be advantageous in several applications. Details of the random access capabilities of the 4D transform mode are reported in.[13]

4. CONCLUSIONS

Image coding standards provide interoperability between codecs built by different manufactures facilitating the development of services and applications. Digital light fields, holograms, and point clouds are novel image modalities providing a sampled representation of a 3D scene, and finding applications in different sectors such as, for example, industry, health-care, gaming. The large amount of data describing the visual information of a scene requires novel coding tools for efficient storage and transmission of these data. This paper presented an overview of the current JPEG standardization efforts towards the design of a generic file format for these image modalities and towards light field image coding tools. The file format enables the mapping of light field, holography, and point cloud information on a common reference grid, defines data structure for the respective metadata, and provides the containers for the compressed imaging bitstreams. The light field coding solutions provides two approaches (tools) for data compression, one exploiting 4D data prediction, the other exploiting a 4D data transform.

ACKNOWLEDGMENTS

The research leading to these results received funding from the Cagliari2020 project (MIUR, PON04a2 00381), the DigitArch Cluster Top-Down project (POR FESR, 2014-2020) and JSPS KAKENHI Grant Number JP17H03267, and the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC Grant Agreement n. 617779 (INTERFERE). The authors also would like to thank the Samsung Reasearch Brazil (SRBR).

REFERENCES

[1] Ebrahimi, T., Foessel, S., Pereira, F., and Schelkens, P., "JPEG Pleno: Toward an efficient representation of visual reality," *IEEE MultiMedia* **23**, 14–20 (Oct 2016).

[2] Perra, C., "Assessing the quality of experience in viewing rendered decompressed light fields," *Multimedia Tools and Applications* **77**, 21771–21790 (Aug 2018).

[3] Perra, C., Astola, P., da Silva, E. A. B., Khanmohammad, H., Pagliari, C., Schelkens, P., and Tabus, I., "Performance analysis of JPEG Pleno light field coding," in [*SPIE Optics + Photonics 2019, Applications of Digital Image Processing XLII*], *Proc. SPIE* **11137**, SPIE (2019).

[4] "Information technology — JPEG 2000 image coding system – Part 1: Core coding system." International Standard ISO/IEC IS-15444-1 (2016).

[5] "Information technology — JPEG 2000 image coding system – Part 2: Extensions." International Standard ISO/IEC IS-15444-2 (2004).

[6] "ISO/IEC JTC 1/SC29/WG1N84065:Information technology - JPEG Pleno Plenoptic image coding system - part 2: Light field coding," (2019).

[7] Astola, P. and Tabus, I., "WaSP: Hierarchical Warping, Merging, and Sparse Prediction for Light Field Image Compression," in [*2018 7th European Workshop on Visual Information Processing (EUVIP)*], 1–6 (Nov 2018).

[8] Sayood, K., [*Introduction to Data Compression*], Morgan Kaufmann, 4th ed. (2012).

[9] Samet, H., "The quadtree and related hierarchical data structures," *ACM Comput. Surv.* **16**, 187–260 (June 1984).

[10] Andrew, J., "A simple and efficient hierarchical image coder," in [*Proceedings of International Conference on Image Processing*], **3**, 658–661 vol.3 (October 1997).

[11] Bell, T. C., Cleary, J. G., and Witten, I. H., [*Text Compression*], Prentice Hall, 1s ed. (1990).

[12] Pereira, F., Pagliari, C., da Silva, E. A. B., Tabus, I., Amirpour, H., Bernardo, M., and Pinheiro, A., "ISO/IEC JTC 1/SC29/WG1N83029:Information technology - JPEG Pleno Light Field Coding Common Test Conditions V3.2, 83th JPEG Meeting, Geneva, Switzerland," (2019).

[13] da Silva, E. A. B., de Carvalho, M. B., Pagliari, C. L., Pereira, F., Pereira, M. P., de Oliveira e Alves, G., Testoni, V., and Garcia, P., "ISO/IEC JTC 1/SC29/WG1M80057: Exploration Studies 1.4 for JPEG Pleno, Study on random access extensions to architecture (4D-DCT) 80th JPEG Meeting, Berlin, Germany," (2018).