IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2020, accepted September 13, 2020, date of publication September 18, 2020,
date of current version September 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024844

The JPEG Pleno Light Field Coding Standard 4D-Transform Mode: How to Design an Efficient 4D-Native Codec

GUSTAVO DE OLIVEIRA ALVES[1], (Member, IEEE), MURILO BRESCIANI DE CARVALHO[2], (Member, IEEE), CARLA L. PAGLIARI[3], (Senior Member, IEEE), PEDRO GARCIA FREITAS[4], ISMAEL SEIDEL[4], (Member, IEEE), MARCIO PINTO PEREIRA[1], (Member, IEEE), CARLA FLORENTINO SCHUELER VIEIRA[1], VANESSA TESTONI[4], (Senior Member, IEEE), FERNANDO PEREIRA[5], (Fellow, IEEE), AND EDUARDO A. B. DA SILVA[1], (Senior Member, IEEE)

[1]PEE/COPPE/DEL/POLI/UFRJ, Universidade Federal do Rio de Janeiro, Rio de Janeiro 21941-901, Brazil
[2]TET/CTC, Universidade Federal Fluminense, Niterói 24220-900, Brazil
[3]PEE/PGED/IME, Instituto Militar de Engenharia, Rio de Janeiro 22290-270, Brazil
[4]Samsung R&D Institute Brazil (SRBR), Campinas 13084-791, Brazil
[5]Instituto de Telecomunicações, Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal

Corresponding author: Carla L. Pagliari (carla@ime.eb.br)

This work was supported in part by the Samsung R&D Institute Brazil (SRBR), in part by CNPq, in part by CAPES, in part by FAPERJ, and in part by the Instituto de Telecomunicações-IT.

ABSTRACT The increasing demand for highly realistic and immersive visual experiences has led to the emergence of richer 3D visual representation models such as light fields, point clouds and meshes. Light fields may be modelled as a 2D array of 2D views, corresponding to a large amount of data, which demands for highly efficient coding solutions. Although static light fields are inherently 4D structures, the coding solutions in the literature mostly employ traditional 2D coding tools associated with techniques such as depth-based image rendering to generate a residual, again to be coded using available 2D coding tools. To address the market needs, JPEG has launched the so-called JPEG Pleno standard, which Part 2 is dedicated to light field coding. The JPEG Pleno Light Field Coding standard includes two coding modes, one based on the 4D-DCT, so-called 4D-Transform mode, and another based on depth-based synthesis, so-called 4D-Prediction. The 4D-Transform coding mode standardizes for the first time a 4D-native light field coding solution where the full light field redundancy across the four dimensions is comprehensively exploited, somehow extending to 4D the 2D coding framework adopted decades ago by the popular JPEG Baseline standard. In this context, this paper describes and analyzes in detail the conceptual and algorithmic design process which has led to the creation of the JPEG Pleno Light Field Coding standard 4D-Transform coding mode. This has happened through a sequence of steps involving technical innovation design and integration where increasingly sophisticated coding tools have been combined and improved to maximize the final rate distortion (RD) performance.

INDEX TERMS JPEG Pleno standard, light field coding, 4D-transform mode, 4D-DCT.

I. INTRODUCTION

The recent developments in visual representation technology and applications have shown that users increasingly ask for highly immersive and realistic 3D experiences, which require new acquisition devices, representation models and coding solutions. Moreover, *a posteriori* functionalities such as refocusing, relighting, and perspective changes are demanded, again highlighting how critical is to adopt more powerful and richer visual representation models. In fact, stereo and multi-view systems fail to produce sufficiently accurate, immersive and realistic user experiences, mainly due to their limited degrees of freedom, low number of views, large camera baselines and the inherent difficulties in the

The associate editor coordinating the review of this manuscript and approving it for publication was Gulistan Raja.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

disparity estimation and synthesis processes. One way to mitigate these weaknesses is by acquiring richer light information from the visual scene, targeting to offer the users the well-known 6-DoF (Degrees of Freedom), where they may exercise displacements in the three rotational and three translational dimensions, thus enjoying visual experiences better matching what would happen in the real world.

A very powerful way to model the light information in the real world is through the so-called *plenoptic function*, which measures the intensity of light at any 3D viewpoint (x,y,z), coming from any angular direction (θ, ϕ), over time (t), and for each wavelength (λ) [1], [2]. If the 7-dimensional plenoptic function information could be effectively captured and replicated, it would be possible to offer the user very immersive and realistic experiences. The recent emergence of more powerful sensors and systems for light information acquisition has singled out as most relevant three key representation models for the plenoptic function visual information, notably light fields, point clouds and meshes. The light field (LF) representation model is based on an ideally large number of 2D scene perspectives with 2D parallax, allowing the light intensity from a single position in the 3D space to be acquired from multiple angular directions/perspectives. On the contrary, the point cloud (PC) and mesh representation models target to directly represent the geometry of the visual objects/scene in the 3D world and are known as geometry-based representation models in opposition to the image-based representation models associated to the LFs. Since these representation models target to achieve high realism and immersion, they have as common feature the large amount of raw data associated, thus critically asking for very efficient coding solutions in order to make affordable transmission and storage enabled applications.

Because still based on the notion of pixel, and not of voxel as point clouds, the LF model tends to look more familiar and thus got much attention in recent years, notably towards the development of efficient coding solutions. In the context of the LF representation approach, it is common to distinguish two scenarios, intimately related to the acquisition process of the 2D views array, notably related to the angular sampling density: the so-called *lenslet cameras* and the *2D arrays of cameras*, disposed or not along a regular planar grid [3]; these approaches are relatively equivalent in practice, with the so-called *camera* or *perspective baseline* being the critical difference. The lenslet cameras include an array of micro-lenses between the main lens and the image sensor, which is able to capture the light rays associated to multiple angles [4]. In a lenslet or LF camera, a micro-lens array, consisting on a set of micro-lenses, is placed at the focal plane of the main lens, this means at a given distance from the photo sensor. Each micro-lens can be understood as a small camera, thus capturing light rays from a specific point in the scene, acquiring a so-called *Micro-Image (MI)* [4]; the MI model represents the LF as a large 2D matrix of MIs and only refers to lenslet camera acquisition. However, the MI model data may be rendered into a 2D array of so-called *Sub-Aperture (SA) images*, each corresponding to a different viewpoint into the visual scene. At this stage, the 2D array of lenslet rendered perspectives becomes equivalent to an acquired 2D array of perspectives captured using regular cameras, notably allowing the usage of the same coding solutions. Naturally, from the acquired perspectives, other (virtual) perspectives into the 3D world may be rendered and some *a posteriori* processing functionalities may be provided. This is possible because a richer representation of the visual scene is available.

The emergence and appeal of LF acquisition and processing technologies, and the potential for more powerful user experiences, has led the Joint Photographic Experts Group (JPEG) committee to launch in February 2015 [5] a new project, called *JPEG Pleno*. It targets the efficient representation of plenoptic imaging, notably LFs, point clouds and holographic data, fulfilling a previously identified set of requirements [6]. Within JPEG Pleno, the LF coding part has been the one with the fastest development, with a Call for Proposals in January 2017 [7], while the point cloud coding activity has launched a Call for Evidence in July 2020 [8] and the holographic data coding activity will launch a Call for Proposals in October 2020 [9]. The request for standard LF coding solutions comes from a wide range of application domains, notably virtual and augmented reality, cultural heritage, sports broadcasting, personal communications, visual surveillance, medical, etc. In this context, JPEG Pleno Part 2 (which Final Draft International Standard (FDIS) has been concluded in April 2020 [10]) standardizes LF coding technologies, considering the LF modelled as a 2D array of 2D views, independently of the acquisition process, and thus of views density and baseline.

Following a three-year development process, the JPEG Pleno Light Field Coding standard includes two LF coding modes, one based on the 4D-DCT, so-called *4D-Transform mode*, and another based on depth-based synthesis, so-called *4D-Prediction mode*. While the two JPEG Pleno coding modes may code any LFs represented as a 2D array of 2D views, the former performs better for more densely angular sampled LFs, such as lenslet LFs, while the latter performs better for more sparsely angular sampled LFs; this performance difference was the key motivation for including both coding modes in the JPEG Pleno Light Field Coding standard. The 4D-Transform coding mode standardizes a 4D-native LF coding solution where the full LF redundancy, across the four LF dimensions, i.e. across and within views, is comprehensively exploited. This coding mode somehow extends to 4D the 2D coding framework that has been adopted decades ago for the very popular JPEG Baseline standard. The 4D-Prediction mode relies on depth maps for the warping process, thus showing an RD performance that is very sensitive to the quality of the depth maps. On the other hand, the 4D-Transform mode does not explicitly rely on any geometric data, which may not be available or may be complex/expensive to obtain. The 4D-Transform mode LF codec, which is rather innovative considering the previous literature, is the target of this paper.

In this context, this paper offers a detailed description and analysis of the design process of the JPEG Pleno Light Field Coding standard 4D-Transform coding mode, notably in terms of concepts and algorithms. This paper will guide the reader along the steps performed in terms of technical innovation design and integration, where successively more sophisticated coding tools have been developed and combined to maximize the final RD performance. This process has brought a rather simple 4D extension of the JPEG Baseline standard to a state-of-the-art LF codec, with top performance for densely angular sampled LFs, while avoiding any dependence on depth maps, which availability and quality are always critical. This in-depth presentation of the original work that has culminated in the JPEG Pleno Light Field Coding standard 4D-Transform coding mode is presented here by the expert team who designed, implemented and tested this codec, with material that has never been published before. In summary, the value of this paper is a deep motivation, description, analysis and assessment of the JPEG Pleno Light Field Coding standard 4D-Transform coding mode by those who shaped it to become the first 4D-native LF coding standard.

To achieve its objectives, this paper is organized as follows. Section II reviews the literature on static LF coding, with a special attention to the JPEG Pleno Light Field Coding standard. Then Section III introduces the general 4D LF coding framework, which has been initially adopted and driven the improvement of the successive LF coding solutions. These successively more sophisticated LF coding solutions are presented in Sections IV, V and VI while Section VII offers a comparative RD performance study regarding relevant LF coding benchmarks under the appropriate JPEG Pleno Common Test Conditions (CTC) [11]. Finally, Section VIII reviews the main conclusions and offers directions for future work.

II. RELATED WORK

LF coding has been extensively researched in the last few years. A wide range of LF coding solutions have been proposed, from well-know, off-the-shelf standard codecs, to those specially designed for LF data. Two surveys on LF field coding are available in [12], [13], which analyze and discuss not only the LF coding solutions themselves but also their associated RD performance. An important aspect to be kept in mind, not directly related to the used technology, but paramount from a market point of view, are the LF codec licensing conditions. This aspect may be so relevant, notably for JPEG standards, that some efficient technologies may be excluded from a LF coding solution as it happened for HEVC in the JPEG Pleno Light Field Coding standard 4D-Prediction mode [10] to adopt instead JPEG 2000, which is less efficient but effectively royalty free.

In [12], a LF coding classification taxonomy is proposed, which is very helpful to organize a review of the relevant literature. Staying in the realm of lossy coding, the first important distinction is between micro-image (MI) and perspective-image (PI) based coding, where MI coding specifically refers to lenslet camera created LF content. The next important distinction regards the type of data involved and differentiates texture-only based LF coding from texture+geometry-based, e.g. depth maps, LF coding; in the latter case, some geometry information has to be specifically acquired or extracted from the texture with the help of some acquisition parameters, the most common being depth and disparity maps; since geometry information may not be always available or available with the required quality, these solutions may not be usable at all or be very sensitive to the depth accuracy in terms of compression performance. Finally, the third key distinction regards the data structure used for the LF coding process, notably: i) *Single 2D Image*, where the LF is arranged into a single 2D image; ii) *Layered Sets of Images*, where the LF is arranged in two or more layered sets of images (corresponding to PIs or MIs); iii) *Pseudo Video*, where the LF is arranged as a 'temporal' sequence of images (corresponding to PIs or MIs) following a specific scanning order; iv) *Pseudo Multi-View Video*, where the LF is arranged as multiple 'temporal' sequences of images (typically corresponding to PIs); and v) *Multi-Dimensional Array of Images*, where the LF is arranged as an N-dimensional array of images.

A. MOST RELEVANT LIGHT FIELD CODING SOLUTIONS

In the following, the most recent and representative static LF coding solutions will be reviewed guided by the taxonomy mentioned above [12]. Starting by the texture-only LF codecs, the simplest type of LF coding solutions is very likely the one based on standard image codecs, which are directly applied to the raw LF data arranged as a (large) single image, either MI- or PI-based. Such image compositions present spatial correlation that may be exploited using suitable well-known intra coding tools; while the redundancy inside the MIs or PIs is naturally exploited by the image codec, the use of appropriate intra prediction tools may also allow exploiting, although not completely, some of the redundancy across MIs or PIs. The LF coding solution in [14] uses the High Efficiency Video Coding standard - Screen Content Coding (HEVC-SCC) extension [15]–[17] to code the LF data organized as a single, large 2D matrix. The Intra Block Copy mode, used in the screen content coding pipeline, proved to be useful for LF coding as, e.g. the MIs, tend to present repetitive patterns. The solution in [18] introduces novel intra coding tools into HEVC, notably involving self-similarity concepts, to further exploit the redundancy across MIs; such coding solutions are more often used for densely angular sampled LFs such as lenslet LFs.

Unlike the above solutions, the pseudo-video coding approach uses standard video codecs to exploit both the spatial redundancy, within MIs or PIs, and the redundancy across them. In this case, prior to coding, the LF data is rearranged as a video sequence using formats able to enhance the correlation among the 'video frames', either PIs or MIs; three video coding standards have been used in [19] to code LF

data in a pseudo-video approach, notably H.264/AVC [20], HEVC [21] and Versatile Video Coding (VVC) [22]. To maximize the RD performance by better exploiting the redundancy between MIs or PIs, multiple video coding prediction structures and scanning patterns have been proposed in the literature for rearranging the LF data into a pseudo-video. More recently, a HEVC-based LF codec, labelled HEVC-HR, has been proposed which mixes hybrid prediction structures for MIs or PIs, depending on which is more efficient for different parts of the LF [23].

Because LF data corresponds to multiple views, it is not surprising that also multi-view coding solutions and standards have been used to code LFs. Such LF coding solutions pre-arrange the views into several pseudo-video sequences to apply a multi-view video codec, notably a standard one like the Multi-View Video Coding (MVC) or MV-HEVC standards. In [24], densely angular sampled LFs are coded by employing a technique proposed in [25], which reduces the redundancy among the PIs by simplifying their SA projection. The exploitation of the spatial and temporal correlations in the pseudo-videos (produced by proper PIs arrangement) is made through MV-HEVC, the HEVC multi-view extension [21]. The coding solution proposed in [26] encodes non-densely angular sampled LFs by partitioning the LF data into key views and the so-called *decimated views*. While the key views are coded using MV-HEVC, the decimated views are predicted using a shearlet transform-based extrapolation, and the generated residual also coded with MV-HEVC. The work in [27] treats the LF as a multi-view sequence coded with MV-HEVC. Virtual reference frames predicted from reconstructed neighbouring frames, using a deep neural network, are used as additional reference frames in a modified MV-HEVC hierarchical coding structure. In [28], MV-HEVC is modified by introducing a hierarchical organization of the input LF views, where each frame is located to a specific level based on its location in the 2D view (PI) matrix.

More recently, there have been successful attempts to design LF coding solutions not based on standard 2D image or video codecs but instead on the full and direct exploitation of the full 4D redundancy present in a LF as a whole. In [29], the authors employ compressive sensing and convolutional neural networks to code a LF as a 4D tensor. Studies on the sparsity of the 4D-DCT coefficients associated to a LF [30], [31] led to the development of a simple LF codec based on the separable 4D-DCT [32], referred to as *MuLE (Multidimensional Light Field Encoder)*; this codec shows good RD performance when there is large inter-view redundancy as for lenslet LFs and it is an early version of the LF codec taken as the main focus of this paper, the JPEG Pleno Light Field Coding 4D-Transform mode.

Differently from all LF codecs above, another class of LF coding solutions are those relying on the availability of some geometry information, notably depth or disparity maps, to perform some kind of prediction using view synthesis; the associated residue is typically coded with an available standard 2D codec. In general, this type of coding solutions heavily depend on the accuracy of the auxiliary information, notably depth maps. In [33], a depth-adaptive convolutional neural network is integrated into the HEVC reference software [21], combining the 35 HEVC intra prediction modes with two newly proposed intra prediction modes. For each HEVC's Coding Unit block size, the two new intra prediction modes select different reference blocks according to the depth of the current block. In [34], a novel coding solution based on Fourier Disparity Layer representation is introduced, where the LF is partitioned into subsets of views (PIs). While the first subset is coded as a video sequence using HEVC [21], the remaining subsets are iteratively estimated using Fourier Disparity Layer representation and the prediction residual coded with HEVC; in this solution, both depth map estimation and view synthesis techniques play important roles. In [35], a depth-based LF codec named *Warping and Sparse Prediction (WaSP)* is proposed; an improved version of this codec has become the 4D-Prediction mode of the JPEG Pleno Light Field Coding standard [10], [36], [37]. The WaSP codec is based on depth-based warping and warped reference views merging, which build the main prediction stage. Selected views (PIs) of the LF are labelled as *reference views*, while the remaining views are labelled as *intermediate views*. The WaSP codec works in a hierarchical order by partitioning the views into disjoint sets with the initial set of reference views corresponding to the lowest hierarchical level. The predicted views for a given hierarchical level are synthesized from the views of the lower hierarchical levels. The texture information and associated depth information for the reference views are encoded using off-the-shelf codecs such as JPEG 2000 [10], [36], [37] or HEVC [38]. A WaSP codec variation, dubbed *Warping and Sparse Prediction on Regions (WaSPR)*, which performs residual coding using HEVC instead of JPEG 2000, together with an improved pre-processing stage, yields significantly better RD performance [38]. As common for other depth-based LF codecs, WaSP and WaSPR offer an RD performance that is quite sensitive to the accuracy of the depth estimation and view synthesis processes. The solution in [39] uses depth-assisted view estimation to recover the entire LF from a small subset of previously coded views; the initially selected reference views are coded with HEVC [21]. The remaining views are predicted using a linear combination of the decoded reference views and the residuals encoded using Principal Component Analysis (PCA), targeting to improve the RD performance; this codec is known as *LF Translational Codec (LFTC)*.

B. JPEG PLENO LIGHT FIELD CODING STANDARD

As mentioned before, Part 2 of the JPEG Pleno standard is dedicated to Light Field Coding and has reached the Final Draft International Standard (FDIS) phase in 2020 [10]. To efficiently address the coding of LFs with different characteristics, notably sampling density, the JPEG committee has decided to standardize two totally independent coding modes, the 4D-Prediction and 4D-Transform coding modes [10], [36], [37], see Fig. 1; both these coding modes are PI-based as

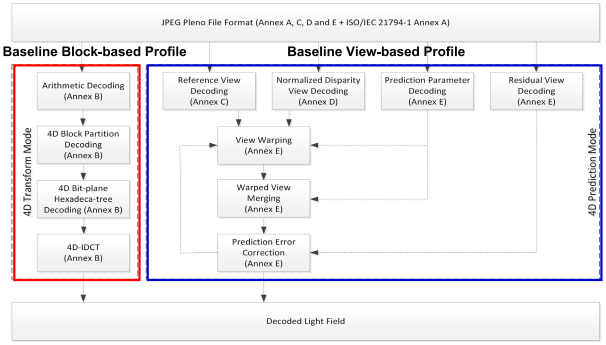


FIGURE 1. Generic JPEG Pleno LF decoder architecture, highlighting the 4D-Transform and 4D-Prediction coding modes, which correspond to two different profiles (adapted from [10]).

this brings more flexibility since also MI-based LFs are commonly consumed as a set of PIs. The motivation to include the two coding modes was related to their respective better performance for sparsely and densely angular sampled LFs.

The 4D-Prediction coding mode performs depth-based rendering for some LF views and is based on the WaSP coding solution detailed in [10], [36], [37] and briefly described in Subsection II-A. This coding mode selects some views, PIs, as reference views, which texture and depth are coded using the (royalty free) JPEG 2000 standard; while more efficient coding solutions could be selected, e.g. HEVC Intra or VVC Intra, JPEG decided for JPEG 2000 due to its strong goal to offer royalty free coding standards, what does not commonly happen for MPEG standards. The remaining views, called intermediate views, are coded by exploiting the sample correspondences between them and the reference views, which are obtained from the necessary depth maps and camera parameters. At this stage, the reference view samples are warped into the intermediate view positions, followed by a prediction phase where the multiple warped views are merged into a full intermediate view using least-squares sense optimal predictors over a set of occlusion-based regions [10], [36], [37].

The 4D-Transform coding mode is based on 4D block-based coding of the 4D LF data [37], does not rely on any depth information, and is an evolution on the MuLE coding solution [32]. This is the LF codec which design progress will be deeply motivated, described, analyzed and assessed in this paper. References [37], [40] and Section VII of this paper compare the RD performance of these two standard LF codecs for densely angular sampled LFs since this is the target LF content for the 4D-Transform coding mode, which is the subject of this paper.

JPEG Pleno Part 1 [41], named *Framework*, defines a very flexible file format, named JPL (Jpeg PLeno), which specifies the signalling syntax for detailed information and contextualization of the streams associated to the plenoptic imaging content streams encapsulated in the file. This file acts like an 'umbrella' as it may contain either different representations of the same content, e.g. LF, point cloud, holograms, to be used at different positions of the application pipeline, and/or different parts of a single plenoptic scene, which are independently coded and recomposed at rendering time. Part 1 [41] provides the 'glue' between the three JPEG Pleno data types as allowing them to 'peacefully' coexist in the same scene; this offers the possibility to adaptively represent complex visual scenes where different parts adopt different data types, e.g. a LF for the background and point clouds for the foreground objects, thus easing interaction.

As shown in Fig. 1, the JPEG Pleno Light Field Coding standard defines two profiles, so-called *Baseline block-based* and *Baseline view-based*, corresponding to the two standardized coding modes. In this context, a LF decoding device may be compliant to any of the two profiles or to the two profiles simultaneously, depending on the type of LF content it is expected to decode. Naturally, nothing prevents coding densely angular sampled LF data with the 4D-Prediction mode and vice-versa, only the RD performance may not be the best. For each profile, four levels have been defined depending on the *Maximum number of samples* and *Maximum block dimension* for the Baseline block-based profile and only on the *Maximum number of samples* for the Baseline view-based profile; here a sample corresponds to a single component value per channel, texture or depth.

III. A 4D-NATIVE STATIC LIGHT FIELD CODING FRAMEWORK

To achieve the key objective of this paper, which is to offer the reader a detailed motivation, description, analysis and assessment of the successive design and integration steps that led to the JPEG Pleno Light Field Coding standard 4D-Transform coding mode, this section presents the 4D-native static LF coding framework. This basic framework has been adopted along the design and integration processes of increasingly sophisticated coding tools, targeting to maximize the final RD performance. This LF coding framework is inspired by the very popular and largely deployed JPEG Baseline image coding standard [42]–[44], which encoder architecture is composed by five main modules, notably block partitioning, a transform, a quantizer, a symbol generator and an entropy encoder. Following the same design philosophy as for the JPEG Baseline standard, the proposed 4D-native static LF coding framework includes the following key modules, see Fig. 2:

- *4D Block Partitioning* – Since the selected transform is a block-based one, such as the DCT in the JPEG Baseline standard, the LF has to be first partitioned into 4D blocks prior to the transform; in this case, the 4D blocks are hyperparallelepipeds (higher-dimensionality parallelepipeds), with specific selected fixed or maximum dimensions, which will be ultimately further partitioned in a dynamic way, depending on the specific LF content.
- *4D-DCT* – Extending the JPEG Baseline standard into the 4D world, the selected transform is naturally the 4D-DCT, which is applied to all 4D blocks.

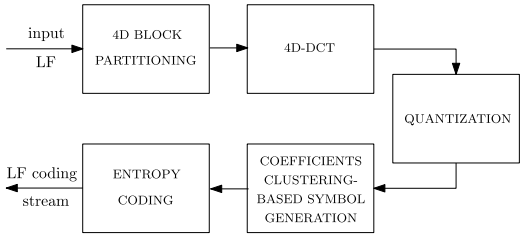


FIGURE 2. 4D-native static LF coding framework.

- *Quantization* – The quantization of the 4D-DCT coefficients is critical to control the target quality and rate; naturally, if appropriate selection criteria are available, different quantization steps may be applied to different 4D blocks and different 4D-DCT coefficients, notably if knowledge is available on their specific subjective quality impact.
- *Coefficients Clustering-based Symbol Generation* – The quantized 4D-DCT coefficients in the 4D hyperparallelepipeds have to be represented as a sequence of symbols for posterior entropy coding. A common solution is to generate coefficients clustering-based symbols, notably to efficiently represent the large groups of coefficients quantized to zero.
- *Entropy Coding* - Finally, some sort of entropy coding is carried out to convert the previously created symbols into a bitstream with as few bits as possible.

It is important to point out that using such a framework, each 4D block is encoded independently of the others, thus generating a bitstream which enables random access to individual 4D blocks at the decoder side.

In the next sections, this simple LF coding framework will guide the LF codec design and integration processes and be filled with successively more sophisticated coding tools, starting from a rather straightforward 4D extension of the JPEG Baseline standard and ending with a novel 4D-DCT based LF codec that is thoroughly 4D-native and has been adopted in the JPEG Pleno Light Field Coding standard as the 4D-Transform mode [10].

IV. MuLE-TSR: STARTING WITH A 4D EXTENSION OF THE JPEG BASELINE CODEC

The JPEG Baseline codec [42] is widely regarded as the most successful image coding standard specified so far; although it has been around for almost 30 years, it is still extremely popular and its usage is still growing. Despite being outperformed by its successors in terms of RD performance, this JPEG codec is still the most used image coding technology and continues to meet the user needs [44] while offering a huge image ecosystem. Therefore, a natural first step in the development of a 4D-native LF codec is to extend the JPEG Baseline image codec [42]–[44] to 4D. Referring to the framework in Fig. 2, a straightforward way to accomplish this is by:

1. Partitioning the LF into non-overlapping 4D blocks of equal dimensions.
2. Computing the separable 4D-DCT of each block instead of the 2D-DCT as in JPEG Baseline. To reduce the 4D-DCT dynamic range requirements, a level-shift operation is performed prior to 4D-DCT computation by subtracting half the dynamic range from all the samples in the 4D block.
3. Performing linear quantization of the 4D-DCT coefficients with the quantization matrix replaced by plain scalar quantization with the same step size for all 4D-DCT coefficients. This simple solution derives from the fact that the visual sensitivity to the 4D-DCT basis functions in LFs is not yet known.
4. Clustering the coefficients using appropriate coefficient scanning followed by run-length encoding (RLE) to generate the symbols to be entropy coded. While RLE can be implemented along the same principles as in JPEG Baseline (see Subsection IV-B), the 2D zig-zag scanning from JPEG Baseline [42] should be replaced by appropriate 4D coefficient scanning (see Subsection IV-A2).
5. Performing entropy coding of the generated RLE symbols.

This proposed 4D extension of the JPEG Baseline codec is referred to as **Mu**ltidimensional **L**ight field **E**ncoder using 4D **T**ransform, coefficient **S**canning and **R**un-length coding (**MuLE-TSR**). From the modules in Fig. 2, those corresponding to Coefficients Clustering-based Symbol Generation and Entropy Coding will be detailed in Subsections IV-A, IV-B and IV-C as defining this MuLE-TSR codec. Those corresponding to the Partitioning, 4D-DCT and Quantization use quite straightforward solutions, and thus will not be further detailed here.

A. 4D-DCT COEFFICIENTS SCANNING

To design efficient 4D scanning patterns, it is appropriate to first analyze the LF energy distribution over the 4D-DCT coefficients; this will be performed for dense angular sampling LFs, such as the lenslet LFs to be used for the assessment of the proposed LF codecs, see Section VII. After this analysis presented in Subsection IV-A1, the design of the 4D scanning patterns will be detailed in Subsections IV-A2.a and IV-A2.b; finally, the proposed 4D scanning patterns are assessed in IV-A2.c in the context of the MuLE-TSR LF codec using the test conditions on Subsection VII-A.

1) 4D-DCT COEFFICIENTS ENERGY DISTRIBUTION FOR DENSELY ANGULAR SAMPLED LFs

Densely angular sampled LFs tend to show a large amount of spatio-angular redundancy [31]. When applying the 4D-DCT to a 4D block, it is expected that most of the data block energy is concentrated in the lower frequency transform coefficients. The key objective of the coefficients scanning process is to generate an ordered list of coefficients such that those with higher energy are scanned before those with lower

energy as this helps improving the final RD performance. To make a proper design of the 4D scanning patterns for LF 4D-DCT coefficients, it is important to first understand the nature of the LF data to be 4D-DCT coded. For this purpose, the energy distribution of the 4D-DCT coefficients is assessed to obtain unique information about the LFs. The computation of the 4D-DCT coefficients energy distribution is carried out through the following steps:

1) *4D block partitioning*: The input LF is partitioned into 4D blocks of dimensions $8 \times 8 \times 8 \times 8$ (while these block dimensions have been chosen because its DCT can be efficiently computed, the specific block dimensions should not affect the overall conclusions);
2) *4D-DCT transform*: A separable 4D-DCT [45] is applied to each 4D block of luminance samples as this is the component with larger weight on the RD performance;
3) *4D-DCT coefficient energy estimation*: The energy of each 4D-DCT coefficient is estimated as the variance computed over all 4D blocks in a LF for each specific 4D-DCT coefficient position [46].

Fig. 3 shows the luminance 4D-DCT coefficients energy distribution for selected LFs from the JPEG Pleno CTC document [11]. While the lenslet LFs, *Bikes* and *Fountain*, are presented in Subsection VII-A, and are part of the set of lenslet LFs used along the rest of the paper for performance assessment, the *Greek* and *Set2 2K sub* LFs are not lenslet LFs. The *Set2 2K sub* LF corresponds to a 33×11 array of 1920×1080 resolution views and shows a table with multiple, fine detail objects of different sizes, placed at different depths, thus presenting medium to high disparity values. The *Greek* synthetic LF corresponds to a 9×9 array of 512×512 resolution views and shows two head statues located farther and closer to the (virtual) camera grid, over a table, and against a slightly textured background, thus presenting medium to large disparities; although the texture is apparently smooth, it is rather noisy. The coordinates (s, t) correspond, respectively, to the horizontal and vertical coordinates of a view in the 2D view array and (u, v), respectively, to the horizontal and vertical coordinates of a sample within a 2D view [2]. In Figs. 3a, 3c, 3e and 3g, the 4D structure is represented as an $u \times v$ 2D array where each element is an $s \times t$ 2D array, while in Figs. 3b, 3d, 3f and 3h it is represented as an $s \times t$ 2D array where each element is an $u \times v$ 2D array. The analysis of Fig. 3 leads to the conclusion that for the lenslet LFs *Bikes* and *Fountain*, the energy decay is much slower along the (u, v) coordinates than along the (s, t) coordinates. Since the faster the 4D-DCT coefficients energy decay along a direction, the larger the redundancy along this direction, these results are coherent with the findings in [31], which state that for lenslet LFs the inter-view redundancy tends to be larger than the intra-view redundancy. On the other hand, for the more sparsely angular sampled LFs *Greek* and *Set2 2K sub*, the energy decay along the (s, t) coordinates tends to be much slower than for the lenslet LFs. For these LFs, the energy of the high frequency 4D-DCT coefficients is still

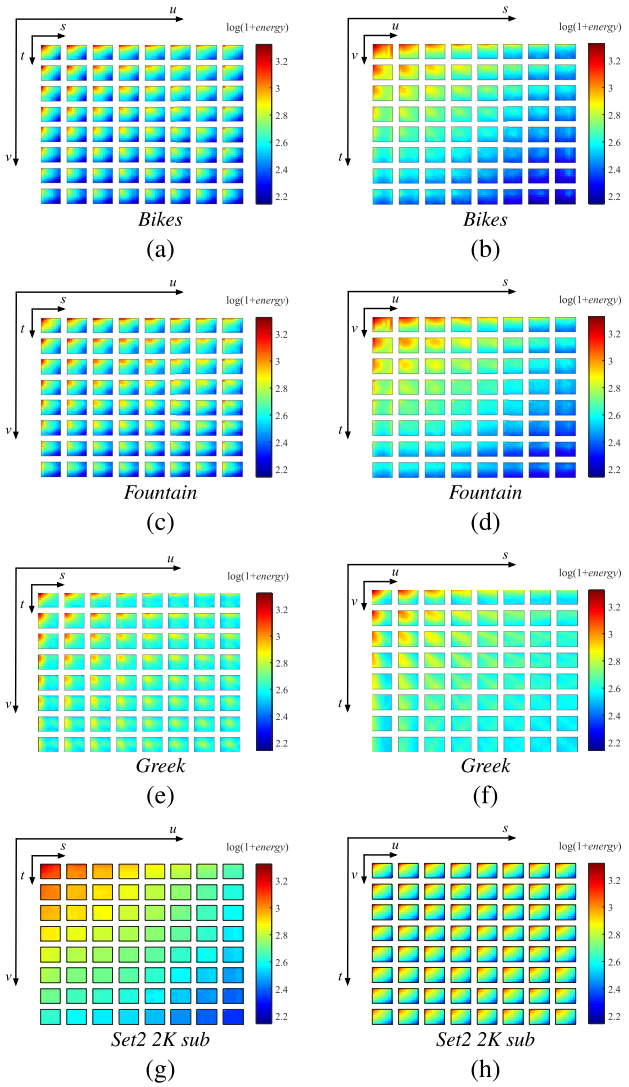


FIGURE 3. Luminance 4D-DCT coefficients log energy distribution for selected LFs from the JPEG Pleno CTC document [11] depicted as: (a), (c), (e) and (g), an $u \times v$ 2D array where each element is an $s \times t$ 2D array; (b), (d), (f) and (h), an $s \times t$ 2D array where each element is an $u \times v$ 2D array.

quite large. This is equivalent to saying that the 4D-DCT energy compaction properties are poorer for such LFs, which is mainly due to the larger baseline. These results are again coherent with the findings in [31]. Such poor energy compaction property of the 4D-DCT for the LFs which angular sampling is more sparse (non-lenslet) is the main reason why the JPEG Pleno 4D-Transform coding mode [37] performs better for lenslet LFs. Since its performance for this type of LFs was the motivation for its inclusion in the JPEG Pleno standard, the remainder of this paper only considers lenslet LFs for assessment purposes.

Thus, for lenslet LFs, it is possible to conclude from Figs. 3a, 3b, 3c and 3d, that a way to boost the probability of having the larger energy 4D-DCT coefficients scanned before the lower energy ones is to have the coefficients (u, v) within each view scanned in an inner loop, while the views (s, t) are

scanned in an outer loop. Although the simple visual inspection already provides a good indication of which may be efficient scanning patterns, it is important to be able to more objectively select the efficient scanning patterns for encoding the LF 4D-DCT coefficients, e.g. using RLE. In the sequel, several 4D-DCT scanning patterns are proposed for selection depending on their impact on the final RD performance.

2) 4D-DCT SCANNING PATTERNS

To design efficient 4D scanning patterns, a good start is to analyze how the 3D diagonal scanning pattern proposed in [47] is derived from the 2D diagonal and zig-zag patterns used in JPEG Baseline [42], H.264/AVC [20], HEVC [21] and VVC [22]. In both the diagonal and zig-zag scanning patterns employed for the 2D-DCT, the coefficients are scanned along a direction perpendicular to the main diagonal of the coefficients' matrix, that is, the direction along which the sum of the coordinates of each coefficient is equal to a constant c. The 3D diagonal scanning pattern for a parallelepiped of coefficients is an extension of the 2D diagonal scanning pattern to 3D where the coefficients are scanned along planes perpendicular to the main diagonal of the 3D parallelepiped [47]. Such planes are characterized by the constraint $x + y + z = c$ on the coefficients coordinates (x, y, z). Fig. 4 depicts a section of the 3D diagonal scanning pattern for a $5 \times 5 \times 5$ block of coefficients, where the highlighted blocks are constrained by $x + y + z = 4$.

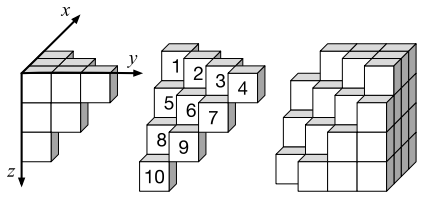


FIGURE 4. 3D diagonal scanning pattern for a 5 × 5 × 5 block, depicting the diagonal plane for $x + y + z = 4$ and corresponding scanning order.

a: 4D DIAGONAL SCANNING PATTERN

As above, the 4D diagonal scanning pattern may be derived from its 3D counterpart by scanning the coefficients along hyperplanes orthogonal to the main diagonal of the 4D hyperparallepiped. These hyperplanes are characterized by the following constraint on the coefficients coordinates: (u, v, s, t): $u + v + s + t = c$. In addition, it is necessary to guarantee that the coefficients on the hyperplane are scanned from the low to the high frequencies, always starting with the DC coefficient. Therefore, an ordered list of the 4D-DCT coefficients corresponding to the 4D diagonal scanning pattern should be generated. This is done by scanning each coordinate independently (in the order u, v, s, t) from the lower to the higher frequencies, while adding to the list only the coordinates satisfying the restriction of belonging to the hyperplane characterized by $u+v+s+t = sum$ (in Step 44vi of Proc 1) as detailed below.

Proc. 1: 4D diagonal scanning pattern order generation

1. *Inputs*:
   - 4D Block dimensions M, N, K and L (u, v, s and t).
2. *Outputs*:
   - *scanningOrder*: The ordered list of coordinates containing the scanning order.
3. *Initialization*:
   - i Variable *maxSum* set to $M + N + K + L$.
   - ii String *scanningOrder* initialized as empty.
   - iii Variables u, v, s and t set to -1.
   - iv Auxiliary variable *sum* set to -1.
4. *Scanning ordering generation*:
   - i. $sum = sum + 1$.
   - ii. $t = t + 1$.
   - iii. $s = s + 1$.
   - iv. $v = v + 1$.
   - v. $u = u + 1$.
   - vi. If $(t+s+v+u) = sum$, append coordinates (t, s, v, u) to the string *scanningOrder*.
   - vii. If $u < M$, go to Step 44v
   - viii. If $v < N$, go to Step 44iv
   - ix. If $s < K$, go to Step 44iii
   - x. If $t < L$, go to Step 44ii
   - xi. If $sum < maxSum$, go to Step 44i
   - xii. Terminate procedure.

Although each coefficient has to be visited $(M + N + K + L)$ times to create the list (Step 44xi of Proc. 1), this does not impact the encoding execution time when adopting this scanning pattern as the sequence of scanning positions can be previously generated offline. This procedure has the advantage to be easily adapted to scanning surfaces other than hyperplanes. For example, if scanning along an hyperspherical surface is desired, it suffices to change the conditional statement in Step 4vi of Proc. 1 to $(t^2 + s^2 + v^2 + u^2) = sum^2$.

b: OTHER PROPOSED 4D-DCT SCANNING PATTERNS

By analyzing Proc. 1, it may be noted that a 3D diagonal scanning pattern can be derived from Proc. 1 by just making one of the dimensions (u, v, s, t) constant (e.g., making $t = t_0$ and dropping the loop on t); also a 2D diagonal pattern can be derived by making two of the dimensions constant (e.g., making $t = t_0$ and $s = s_0$ and dropping the loops on s and t). Based on these 4D, 3D and 2D diagonal patterns, the full set of 4D scanning patterns is proposed as follows (summarized in Table 1):

- 4D diagonal scanning pattern along the t, s, v and u directions; this is exactly the pattern defined by Proc. 1 (*4D diagonal* pattern).
- Four scanning patterns corresponding each to one of the four combinations of three dimensions along which a 3D diagonal scan is performed in an inner loop, with

the fourth dimension being scanned in an outer loop (*3D diagonal + 1D* patterns).
Example: A 3D scan along the s, v an u dimensions performed in an inner loop with a scan along the t dimension in an outer loop (pattern 36).

- Six scanning patterns corresponding each to one of the six combinations of two dimensions, along which a 2D diagonal scan is performed in an inner loop, with the two remaining dimensions being scanned using another 2D diagonal scan in an outer loop (*Double 2D diagonal* pattern).
Example: A 2D diagonal scan along the v and u dimensions in an inner loop and another 2D diagonal scan along the t and s dimensions in an outer loop (pattern 46).
- 12 scanning patterns corresponding to each one of the six combinations of two dimensions, along which a 2D diagonal scan is performed in the innermost loop, with the two remaining dimensions being scanned successively in two outer loops (*2D diagonal + 2D successive* patterns).
Example: A 2D diagonal scan along the t and u dimensions in the innermost loop, an outer loop along the s direction and the outermost loop along the v direction (pattern 27).
- 24 scanning patterns along which each dimension is scanned successively corresponding precisely to the $C_4^2 = 24$ possible different orderings of the four dimensions (*4D successive* patterns).
Example: A scan along the u dimension in the innermost loop, an outer loop along the s dimension, an ever outer loop along the t dimension, a scan along the v dimension in the outermost loop (pattern 15).

As previously pointed out, the inspection of Fig. 3 suggests that efficient scanning patterns for lenslet LFs are those in which the (u, v) dimensions are scanned in an inner loop and the (s, t) dimensions in an outer loop. This type of scanning is provided by pattern 46 in Table 1, and belongs to the class of

TABLE 1. Summarized description of the proposed 4D-DCT coefficient scanning patterns and respective assigned indexes. The order shown is from the inner (1st) to the outer loop.

Mode	Scanning Order			
4D successive				
—	1st	2nd	3rd	4th
0	t	s	v	u
1	t	s	u	v
2	t	v	s	u
3	t	u	s	v
4	t	v	u	s
5	t	u	v	s
6	s	t	v	u
7	s	t	u	v
8	v	t	s	u
9	u	t	s	v
10	v	t	u	s
11	u	t	v	s
12	s	v	t	u
13	s	u	t	v
14	v	s	t	u
15	u	s	t	v
16	v	u	t	s
17	u	v	t	s
18	s	v	u	t

Mode	Scanning Order			
4D successive				
—	1st	2nd	3rd	4th
19	s	u	v	t
20	v	s	u	t
21	u	s	v	t
22	v	u	s	t
23	u	v	s	t
2D diagonal + 2D successive				
—	2D diagonal		3rd	4th
24	vu		t	s
25	su		t	v
26	sv		t	u
27	tu		s	v
28	tv		s	u
29	ts		v	t
30	vu		s	t
31	su		v	t
32	sv		u	t
33	tu		v	s
34	tv		u	s
35	ts		u	v

Mode	Scanning Order	
3D diagonal + 1D		
—	3D	4th
36	svu	t
37	tvu	s
38	tsu	v
39	tsv	u
4D diagonal		
—	4D	
40	tsvu	
Double 2D diagonal		
—	Inner Diag.	Outer Diag.
41	ts	vu
42	tv	su
43	sv	tu
44	tu	sv
45	su	tv
46	vu	ts

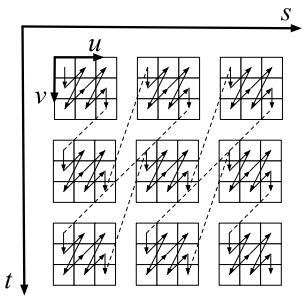


FIGURE 5. Example of Double 2D diagonal scanning pattern for a hyperparallelepided of dimensions $(t, s, v, u) = (3, 3, 3, 3)$: inner loop in (v, u) and outer loop in (t, s).

Double 2D diagonal patterns. Fig. 5 illustrates this scanning pattern.

c: 4D SCANNING PATTERNS ASSESSMENT

The proposed 4D scanning patterns can be assessed by comparing their energy compactness capabilities, corresponding to how fast the energy accumulates as the scanning proceeds. Assuming that the 4D-DCT coefficients' energies have been estimated, the assessment of each 4D scanning pattern is based on the following steps:

1. *4D scanning:* The 4D-DCT coefficients are ordered according to each proposed 4D scanning pattern;
2. *Cumulative energy (CE) computation:* As the ordered 4D-DCT coefficients are visited according to the given 4D scanning sequence, their percent cumulative energy is computed;
3. *Integral of cumulative energy (ICE) computation:* The percent cumulative energies along the scanning path are added for each pattern, what is equivalent to computing the integral/area under the curve of the cumulative energy versus the number of 4D-DCT coefficients already scanned. Since the first cumulative energy has only the contribution of the first coefficient, the second one the contributions of the first and second coefficients, and so on, then the cumulative energy CE_k for the k-th scanned coefficient contributes $N - k + 1$ times for the ICE computation, leading to the expression in Eq. (1):

$$\mathrm{ICE} = \frac{\sum_{k=1}^{N}(N-k+1)\mathrm{CE}_k}{\sum_{k=1}^{N}\mathrm{CE}_k}. \tag{1}$$

From Equation (1), the 4D scanning patterns for which the coefficients with larger energies CE_k are scanned earlier (smaller k) generate a CE curve growing faster, yielding larger ICE values.

Fig. 6a shows the cumulative energy (CE) curve (%) and Fig. 6b shows the corresponding integral of cumulative energy (ICE) for all the proposed scanning patterns for the *Bikes* and *Fountain* LFs. The results in Fig. 3 show that the successive scanning patterns with inner loop in the (u, v) dimensions achieve higher ICE values than the scanning

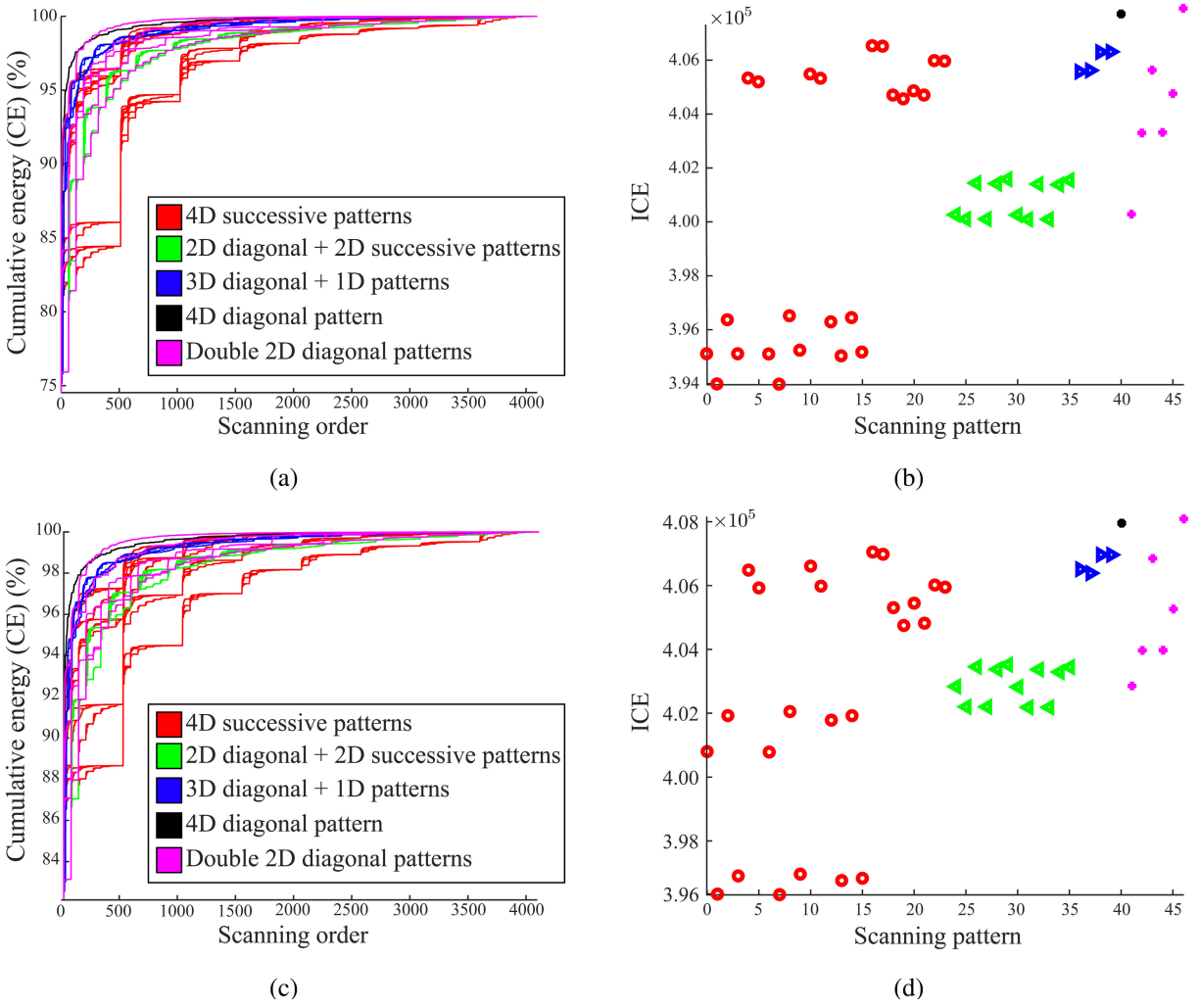


FIGURE 6. Percent cumulative energy (CE) versus scanning order: (a) *Bikes*; (c) *Fountain*; Integral of Cumulative Energy (ICE) for the 47 proposed scanning patterns: (b) *Bikes*; (d) *Fountain*.

patterns with inner loop in the (s, t) dimensions for these LFs. Indeed, for these LFs, the best scanning pattern is the one with index 46, corresponding to the Double 2D diagonal pattern with inner loop in (u, v) and outer loop in (s, t). It has been verified that the behaviour for the other lenslet LFs listed in the JPEG Pleno CTC document [11] is the same.

B. 4D-DCT COEFFICIENTS RUN-LENGTH ENCODING

As discussed above, the output of the 4D scanning process is a 1D array of quantized 4D-DCT coefficients, which must be efficiently encoded, e.g. using RLE. Actually, there is no fundamental difference between the 1D array of scanned 2D-DCT coefficients input to the JPEG Baseline RLE codec [42]–[44] and the one generated by the 4D scanning process described above. Likewise, there should be also no fundamental difference between the RLE encoding processes for the two codecs. In brief, the JPEG Baseline RLE codec scans this linear array and counts the number of zeros before each non-zero coefficient, thus generating a pair (*run*, *bit-depth*) in which *run* represents the count of zeros before the next non-zero coefficient and *bit-depth* corresponds to the number of bits necessary to represent the value of the respective quantized coefficient using Variable Length Integer (VLI) entropy coding. Each symbol is composed by two tokens: the first is the (*run*, *bit-depth*) pair and the second is the *amplitude*, which is the string of symbols corresponding to the binary representation of the coefficient value using *bit-depth* bits. The (*run*, *bit-depth*) pairs, as well as the *amplitude* bits are encoded using an entropy encoder.

The main difference between the two RLE processes mentioned above is basically the type of block it refers, this means 2D versus 4D. Although this difference does not affect how the *amplitude* is encoded, it does affect two important RLE parameters, namely the maximum bit-depth of a transform coefficient, $bit\text{-}depth_{\max}$, and the maximum run of zeros allowed within a block, $run_{\max}$. In what follows, the estimation of these two RLE parameters is detailed.

1) MAXIMUM BIT-DEPTH OF A 4D-DCT COEFFICIENT

The 4D-DCT coefficient value is encoded as the *amplitude* using *bit-depth* bits for VLI entropy coding. The maximum bit-depth for the 4D-DCT coefficients depends on the quantization step size (q_{step}) and the dynamic range of the transform coefficients which, in turn, depend on the dynamic range of the input LF samples and the transform block size. Here, this parameter will be estimated based on the fact that the 4D-DCT coefficients absolute values are not expected to exceed the value of the DC coefficient of a constant 4D block with all samples equal to the maximum value. The input bit-depth for the JPEG Pleno LFs [11] is 10 bits and, therefore, the maximum sample value is $p_{\max} = 1023$. The DC coefficient of such 4D block is given by

$$\mathrm{DC}_{\max} = \mathrm{DC}(T_{4D}(M_{\max})), \tag{2}$$

where $M_{\max}$ is the 4D array with all its entries equal to $p_{\max}$, T_{4D} is the operator applying the 4D-DCT and the operator DC selects the coefficient at position (0, 0, 0, 0) of the 4D matrix. If the 4D-DCT coefficients are quantized with a step q_{step}, the maximum bit-depth of a transform coefficient is given by Eq. (3):

$$bit-depth_{\max} = \log_2(1 + \mathrm{DC}_{\max}/q_{step}). \tag{3}$$

2) MAXIMUM RUN OF ZEROS ALLOWED IN A 4D BLOCK

The maximum run of zeros in a 4D block of DCT coefficients is obtained when one has only two non-zero coefficients that are maximally separated according to the scanning order. The scanning starts with the DC value, that is always encoded; as this corresponds to a run with no zeros before, the maximum run occurs when the second coefficient is the last AC coefficient in the block. Therefore, for a block of dimensions $t \times s \times v \times u$, this maximum run is equal to $tsvu - 2$. However, even for moderately small 4D block dimensions, this value is very high. For example, for $(t, s, v, u) = (13, 13, 31, 25)$, the maximum run of zeros would be 4423, and such a value would result in a very high number of possible (*run*, *bit-depth*) pairs to encode, which would negatively impact the entropy coding performance. Fortunately, although theoretically possible, runs of zeros as large as these are very unlikely. This implies that the $run_{\max}$ value that would enable efficient coding of the (*run*, *bit-depth*) pairs tends to be much smaller than this maximum value.

While the choice of $run_{\max}$ does not interfere on the quality of the decoded LF, it impacts the final rate and, thus, the RD performance. Extensive simulations have been carried out by performing a grid search for $run_{\max}$, the product of 4D block dimensions B_d and the quantization step size, always computing the rate obtained for each coded LF. For all LFs considered, it was observed that, if the rate obtained is allowed to be within 10% of the minimum rate, the corresponding $run_{\max}$ is independent of the quantization step size and is related to the product B_d of the 4D block dimensions by Eq. (4) below:

$$\log_{10}(run_{\max}) = 0.47 \log_{10}(B_d) + 0.34. \tag{4}$$

In this context, the above equation has been used to set $run_{\max}$ for all simulations shown in the sequel.

C. ENTROPY CODING

Entropy coding in MuLE-TSR is performed using an arithmetic encoder [48], [49] with two probability models, one for each token associated to the symbols generated by the RLE. The first model may be either fixed or adaptive, and is used to code the (*run*, *bit-depth*) token. This is done by associating each (*run*, *bit-depth*) token to an index computed as

$$index = run + bit-depth \cdot run_{\max}, \tag{5}$$

which is encoded with the arithmetic encoder. The index of value $bit-depth_{\max}(run_{\max} + 1)$ is associated with the *EOB* (End-of-Block) marker [42]–[44], that accounts for an extra symbol.

The second model is a binary model with fixed equiprobable distribution and is used to further encode the VLI encoded bits of the *amplitude* token.

D. MuLE-TSR RD PERFORMANCE ASSESSMENT

The MuLE-TSR RD performance is assessed using the experimental conditions described in Section VII derived from the JPEG Common Test Conditions (CTC) [11]. For this performance assessment study, the tested 4D-DCT transform dimensions are $(t, s, v, u) = (13, 13, 31, 25)$ and the RD performance is computed for at least one pattern of each of the types of scanning patterns as listed in Table 1, taking care that representative patterns according to the ICE values (see Fig. 6) are chosen, notably:

- *4D successive:* pattern 0 (*tsvu*) and pattern 23 (*uvst*).
- *2D diagonal + 2D successive:* pattern 30 (2D diagonal *uv* and successive *st*).
- *3D diagonal:* pattern 37 (3D diagonal *tvu* and then *s*)
- *4D diagonal:* pattern 40.
- *Double 2D diagonal:* pattern 41 (double diagonal with inner *ts* and outer *vu*) and pattern 46 (double diagonal with inner *vu* and outer *ts*).

The RD performance was assessed for six RD points corresponding to the quantization step parameter values of [0.08, 0.5, 1.3, 4, 7, 20]. Table 2 summarizes the MuLE-TSR BD-Rate [50] results regarding the JPEG Pleno HEVC anchor as defined in [11], using the scanning patterns listed above. The BD-Rate results are coherent with the ICE results shown in Figs. 6b and 6d as the best RD performance is achieved

TABLE 2. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-TSR with different scanning patterns.

	Bikes	*Danger*	*Pillars*	*Fountain*
Pattern 0	9.38	-0.49	12.40	39.49
Pattern 23	0.16	-10.95	3.26	34.33
Pattern 30	9.66	-2.01	12.69	41.33
Pattern 37	11.47	0.41	17.81	41.74
Pattern 40	8.17	-1.87	18.10	40.22
Pattern 41	5.37	-6.20	10.28	39.93
Pattern 46	**-2.14**	**-14.31**	**0.54**	**32.36**

for the scanning patterns 46 and 23, which show higher ICE values. The majority of scanning patterns tested offer BD-Rate savings compared to scanning pattern 0, which is associated to a lower ICE value. The conclusion is that ICE is useful to estimate the RD performance associated to the scanning patterns when used individually.

TABLE 3. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-TSR and the 4D-Prediction mode (WaSP) [51].

	Bikes	*Danger*	*Pillars*	*Fountain*
4D-Prediction mode (WaSP)	1.43	-13.15	**-10.07**	**-4.09**
MuLE-TSR Pattern 46	**-2.14**	**-14.31**	0.54	32.36

Table 3 summarizes the BD-Rate results for MuLE-TSR and WaSP still using the JPEG Pleno HEVC anchor as reference [11]; here the scanning pattern 46 has been used for MuLE-TSR since this was the one with the best BD-rate performance in Table 2. It is observed that the MuLE-TSR RD performance is superior to the JPEG Pleno HEVC anchor [11] for the *Bikes* and *Danger* LFs (negative BD-Rates) and inferior for the *Pillars* and *Fountain* LFs (positive BD-Rates). Clearly, the worst performance happens for the *Fountain* LF for several reasons, notably: i) the object/fountain in the scene is extremely close to the capture device (in this case, the LF camera), thus presenting large clusters of small depth values, not present in the other tested LFs; ii) these small depth values lead to low inter-view redundancy; iii) the object in the scene has a complex texture, which also leads to low intra-view redundancy; and finally iv) the water in the scene/fountain corresponds to a non-Lambertian region, which impairs the overall 4D redundancy. Regarding the 4D-Prediction mode (WaSP) [51], MuLE-TSR performs better for the *Bikes* and *Danger* LFs. However, it is worth noting that a codec such as MuLE-TSR, designed as a simple 4D extension of the JPEG Baseline codec, produces already competitive RD performance results compared to solutions that are considerably more sophisticated; nevertheless, it is clear that the MuLE-TSR LF codec needs major improvements to outperform the JPEG Pleno 4D-Prediction mode for lenslet LFs and be able to 'force its entrance' in the JPEG Pleno Light Field Coding standard as it later happened.

V. MuLE-TSB: IMPROVING BY REPLACING RUN-LENGHT CODING BY BINARY-TREE BIT-PLANE CLUSTERING

The RD performance obtained for the MuLE-TSR codec shows the potential of using a 4D-DCT based approach for LF coding; this potential can be further explored by improving the transform coefficients coding process as it will be proposed in this section. Referring to Fig. 2, the Coefficients Clustering-based Symbol Generation module, which consisted in 4D scanning followed by RLE in MuLE-TSB, is improved by replacing the RLE by a binary-tree partitioning process combined with bit-plane clustering. This process starts with the binary representation of the quantized 4D-DCT coefficients using a bit-depth corresponding to the used quantizer, which is determined using an RD criterion. The coefficient bit-planes are clustered using a binary-tree partitioning. Still referring to Fig. 2, the entropy coding, which encodes now the symbols defining the binary-tree structure, becomes an adaptive binary arithmetic encoder. The remaining blocks in Fig. 2, namely the 4D Block Partitioning, the 4D-DCT itself and the Quantization remain unchanged. The resulting coding solution is named as **Mu**ltidimensional **L**ight field **E**ncoder using 4D **T**ransforms, **S**canning and **B**inary-tree-based bit-plane Clustering (**MuLE-TSB**). Following the LF coding framework in Fig. 2, Subsections V-A and V-B will present the novel binary-tree-oriented bit-plane coding and posterior entropy coding.

A. BINARY-TREE-ORIENTED BIT-PLANE CLUSTERING

The binary-tree-oriented bit-plane clustering aims at locating, in the 4D-DCT coefficients scanned array, those coefficients which are quantized as non-zero. The central idea is to segment the 1D array of 4D-DCT coefficients into segments (S_i) that either have all their coefficients quantized as zero or have length 1, that is, contain just one coefficient (see top of Fig. 7). Note that segment S_5 in Fig. 7, although having a non-zero coefficient, is quantized as all zeros, which is indicated by the z symbol. The binary-tree structures, together with the bit-plane representation, are used for the efficient signalling of such segmentation as well as for the efficient encoding of the coefficients quantized as non-zero.

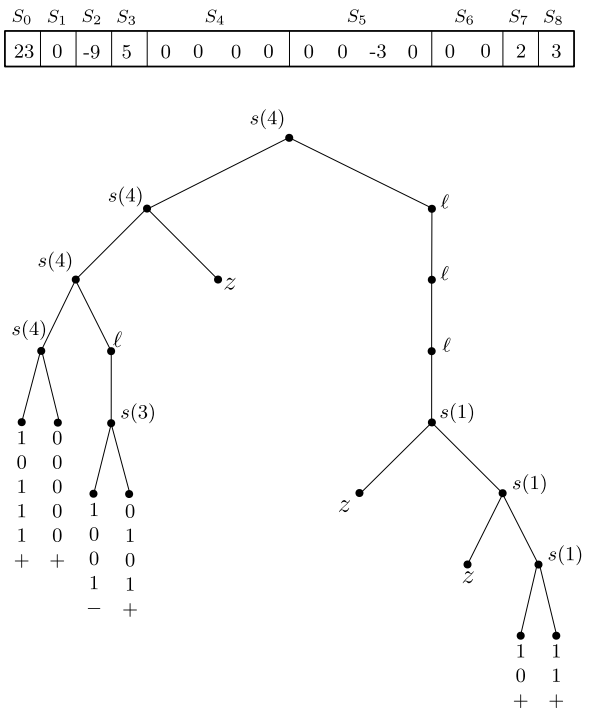


FIGURE 7. Coding of the array in strand (7). The binary-tree encodes the segmentation of the array into S_0 to S_8, what is signalled by ℓ, *s* and *z*, corresponding to the *lowerBitplane*, *splitBlock* and *zeroBlock* flags, respectively. *s*(*n*) means that the segment is split at bit-plane *n*. The corresponding code string is *ssss*10111+00000+ℓ*s*1001−0101+*z*$\ell\ell\ell$*szszs*10+11+.

In this paper, the bit-planes are defined as follows: the lowest bit-plane contains the least significant coefficient bits, and the highest bit-plane the most significant ones. Using this convention, a coefficient is considered non-significant on a given bit-plane if all its bits belonging to higher bit-planes are equal to zero; otherwise, it is considered significant. The highest bit-plane in the 1D coefficient array, $bp_{\max}$, is set so that the largest of the 4D-DCT coefficient magnitudes is smaller than $2^{bp_{\max}+1}$ and larger than or equal to $2^{bp_{\max}}$. The lowest bit-plane, $bp_{\min}$, corresponds to the desired quantization level for the 4D-DCT coefficients, since to represent a coefficient up to a bit-plane $bp_{\min}$ is equivalent to quantize it with a step size equal to $2^{bp_{\min}}$. The bit-planes are scanned from the highest to the lowest one.

For a given segment of the 1D array resulting from the 4D-DCT coefficients scanning, a ternary flag is used to signal the following three possibilities:

- The given segment contains at least one significant coefficient at the current bit-plane and will be split into two segments; this corresponds to the flag *splitBlock*.
- The given segment does not contain any significant coefficient at the current bit-plane and the current bit-plane is decreased without the segment being split; this corresponds to the flag *lowerBitplane*.
- The given segment contains at least one significant coefficient at the current bit-plane but will be discarded (encoded as all zeros) rather than split; this corresponds to the flag *zeroBlock*.

When a 1D segment of dimension N is split, the dimensions of the resulting segments are, respectively, $\lfloor N/2 \rfloor$ and $N - \lfloor N/2 \rfloor$, where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x. The decisions of splitting or not a segment containing significant coefficients are RD optimized, that is, the binary-tree that orients the segmentation of the 1D array of 4D-DCT coefficients is constructed by minimizing the unconstrained Lagrangian cost given by

$$J = D + \lambda R, \tag{6}$$

where R is the rate for coding a segment of coefficients, D is the squared error distortion relative to the original segment of transform coefficients and $\lambda \geq 0$ is a Lagrangian multiplier that controls the compromise between the rate and distortion.

Suppose, for example, that a 1D coefficient array A of dimension 16, resulting from the scanning of a $2 \times 2 \times 2 \times 2$ block of coefficients, is as in the strand below:

$$A = [23\ 0\ {-9}\ 5\ 0\ 0\ 0\ 0\ 0\ 0\ {-3}\ 0\ 0\ 0\ 2\ 3]. \tag{7}$$

The goal of the binary-tree-oriented bit-plane clustering is to quantize and segment A as exemplified at the top of Fig. 7. The binary-tree coding of this segmentation is uniquely specified, as pointed out above, by a sequence of ternary flags with values *lowerBitplane* (ℓ), *splitBlock* (s) and *zeroBlock* (z). The segments with length larger than 1, namely S_4, S_5 and S_6, will have all its coefficients quantized as zero, what is signalled with the *zeroBlock* (z) flag. A coefficient contained in the segments having length 1, namely S_0, S_1, S_2, S_3, S_7 and S_8, should have its bits, from the current bit-plane bp_c down to the one at bit-plane $bp_{\min}$, as well as its sign, forwarded to the entropy encoder together with the sequence of ternary flags.

In this example, since the magnitude of the maximum coefficient is 23, the maximum bit-plane $bp_{\max}$ is set to 4, since $2^4 \leq 23 < 2^5$. Also, the minimum bit-plane is assumed to be $bp_{\min} = 0$. The bit-planes of the non-zero coefficients in the strand (7) are given in Table 4.

TABLE 4. Bit-planes of the non-zero coefficients from the array *A* in strand (7).

coef.	Bit-plane 4	3	2	1	0	sign
23	1	0	1	1	1	+
−9	0	1	0	0	1	−
5	0	0	1	0	1	+
−3	0	0	0	1	1	−
3	0	0	0	1	1	+
2	0	0	0	1	0	+

Fig. 7 also depicts the binary-tree generated when encoding the 1D array of coefficients in strand (7), together with the ternary flags specifying it. The string of symbols *codeString*, encoding the array in strand (7) according to the tree in Fig. 7, is given by

$$\begin{aligned} &codeString \\ &= ssss10111{+}00000{+}\ell s1001{-}0101{+}z\ell\ell\ell szszs10{+}11{+} \end{aligned} \tag{8}$$

The binary-tree-oriented bit-plane clustering of the block of 4D-DCT coefficients is obtained by recursively subdividing a linear array of scanned coefficients until all coefficients quantized as non-zero belong to a length-1 segment. The recursive procedure that generates the *codeString* with the corresponding sequence of ternary flags, bits and signs is given below.

Proc. 2: Recursive binary-tree-oriented bit-plane clustering

1. *Inputs*:
   - *Segment*: A 1D coefficient array of dimension N to be encoded.
   - A Lagrange multiplier λ to be used in the computation of the coding costs according to Eq. (6).
   - *codeString*: A global variable containing the string of symbols to be forwarded to the entropy encoder.
2. *Outputs*:
   - Global variable *codeString* with coding symbols appended.
   - Coding cost, J, for *Segment*.

3. *Initialization*:
   i. $bp_{\max}$ is set so that the largest of the magnitudes of the coefficients in *Segment* is smaller than $2^{bp_{\max}+1}$ and larger than or equal to $2^{bp_{\max}}$.
   ii. The current bit-plane, indexed by bp_c, is set to $bp_{\max}$.
   iii. $bp_{\min}$ is set to minimize the Lagrangian cost according to the steps:
      a) For bp varying from 0 to $bp_{\max}$, execute Steps 33iii3(iii)b to 33iii3(iii)d, and then go to Step 33iii3(iii)e.
      b) The coefficients in *Segment* are quantized with step size 2^{bp}.
      c) D_{est} is computed as the *Segment* distortion per coefficient.
      d) R_{est} is computed as the entropy of the bits of the coefficients not quantized as zero in Step 33iii3(iii)b, taken from the most significant bit to bp.
      e) $bp_{\min}$ is chosen as the bp corresponding to the smallest estimated Lagrangian cost $J_{\text{est}} = D_{\text{est}} + \lambda R_{\text{est}}$.
   iv. The costs J_s of splitting *Segment* and J_z of quantizing *Segment* as all zeros are set to infinity.
4. *Recursive clustering*:
   i. If the magnitudes of the coefficients in *Segment* are all smaller than 2^{bp_c}, then:
      a) Step 4 is recursively called with current bit-plane variable bp_c decremented by 1.
      b) The flag *lowerBitplane* is appended to *codeString*.
      c) If $bp = bp_{\min}$, then the flag *zeroBlock* is appended to *codeString*. The coding cost J is set to $D_z + \lambda R_z$. The distortion D_z is obtained when all coefficients in *Segment* are quantized as zero, and R_z is the rate that would be associated with the transmission of the flag *zeroBlock*; then the procedure terminates.
      d) The procedure returns the coding cost returned from the recursive call performed in Step 4 4i4(i)a and terminates.
   ii. If the magnitude of any coefficient in *Segment* is larger than or equal to 2^{bp_c}, then:
      a) If $N = 1$, that is, *Segment* consists of a single coefficient,
         a1) Set the coding cost J_s to the cost of this single coefficient.
         a2) Append to *codeString* the bits of the magnitude of this single coefficient, from the current bit-plane bp_c to $bp_{\min}$, followed by the coefficient sign, and terminate the procedure.
      b) Set the coding cost J_z to $D_z + \lambda R_z$. The distortion D_z is obtained when all coefficients in *Segment* are quantized as zero, and R_z is the rate that would be associated with the transmission of the flag *zeroBlock*.
      c) *Segment* is split into two sub-segments $Segment_1$ and $Segment_2$, of dimensions $\lfloor N/2 \rfloor$ and $N - \lfloor N/2 \rfloor$, respectively. Then the procedure in Step 4 is recursively called for both $Segment_1$ and $Segment_2$. The coding costs returned by these two calls are added to form the cost J_s.
      d) If $J_z < J_s$, set the coding cost J to J_z and append the flag *zeroBlock* to *codeString*.
      e) If $J_s \leq J_z$, set the coding cost J to J_s and append the flag *splitBlock* to *codeString*.
      f) The coding cost J is returned and the procedure terminates.

It is interesting to observe that, without Lagrangian optimization (e.g., taking the Lagrangian multiplier as zero), the proposed binary-tree-based bit-plane clustering scheme would resemble the quadtree-oriented clustering solution adopted in [52] for wavelet image coding. In Section VI, the binary-tree-oriented bit-plane clustering described here will be extended from 1D to 4D and named hexadeca-tree-oriented bit-plane clustering; this will be a key building block of the MuLE-MTH codec, a fully 4D-native light field codec, the most sophisticated in the MuLE family.

B. ENTROPY CODING

The binary-tree-oriented bit-plane clustering procedure above returns the lowest Lagrangian cost and the resulting associated coding string, which defines the optimal binary-tree. The 4D coefficients, ternary flags, and probability context information (see below), generated during the encoding process, are input to an Adaptive Binary Arithmetic Coder (ABAC), which generates the coded representation for the input 4D block. ABAC requires that for the models with three symbols, such as the binary-tree segmentation ternary flags, each symbol is represented as two binary words. The data is coded into the bitstream as follows:

- *Transform coefficient signs:* A fixed probability model with two symbols is used for the signs;
- *Transform coefficient magnitude bits:* Different contexts are used depending on which bit of the coefficient magnitude is being encoded, one context corresponding to each bit-plane, thus resulting in a total of 32 contexts; this is because the dynamic range of the coefficients is assumed to be such that their magnitudes belong to the interval $[0, 2^{32})$; adaptive probability models are used;
- *Binary-tree flags:* The partition flags *lowerBitplane*, *splitBlock* and *zeroBlock* are input to the arithmetic coding as the binary words '00', '01' and '10', respectively. The context used depends on the current bit-plane bp and on whether the first or the second bit is being encoded; this results in a total of 64 contexts. Again, adaptive probability models are used.

The arithmetic encoder contexts are reset for every 4D block encoded so that the blocks are independently encoded, and thus MuLE-TSB offers 4D block-level random access.

C. MuLE-TSB RD PERFORMANCE ASSESSMENT

The MuLE-TSB RD performance assessment is made under the same JPEG Pleno CTC [11] as before, described in Section VII-A for 4D transform dimensions fixed to $(t, s, v, u) = (13, 13, 31, 25)$. As for MuLE-TSR, the RD performance is first assessed for the same scanning patterns as in Table 2. Table 5 shows the BD-Rate gains for the selected scanning patterns, taking as reference the JPEG Pleno HEVC anchor [11]. The best RD performance is obtained with scanning pattern 23, followed by scanning patterns 46 and 40, while the worst results are obtained for patterns 0 and 30. Once again, the RD performance is coherent with ICE results discussed in Section IV. Rate savings up to 59.95% (Table 5) are obtained when comparing the RD performance for patterns 23 and 0. By comparing the values in Table 5 with the corresponding values in Table 2, it may be concluded that for MuLE-TSB the scanning pattern choice has a larger impact on the RD performance than for MuLE-TSR, mostly due to the more efficient clustering of the 4D-DCT coefficients provided by the binary-tree-oriented bit-plane clustering.

TABLE 5. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-TSB with different scanning patterns.

	Bikes	*Danger*	*Pillars*	*Fountain*
MuLE-TSB Pattern 0	33.35	27.63	49.48	64.73
MuLE-TSB Pattern 23	**-8.20**	**-20.38**	**-10.47**	**26.57**
MuLE-TSB Pattern 30	35.28	27.80	53.46	69.93
MuLE-TSB Pattern 37	25.43	19.58	40.99	56.33
MuLE-TSB Pattern 40	3.83	-3.84	20.09	38.05
MuLE-TSB Pattern 41	22.73	15.73	40.06	53.58
MuLE-TSB Pattern 46	-0.65	-9.73	1.66	32.57

TABLE 6. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11], for MuLE-TSB, MuLE-TSR, and JPEG Pleno 4D-Prediction mode (WaSP).

	Bikes	*Danger*	*Pillars*	*Fountain*
4D-Prediction mode (WaSP)	1.43	-13.15	-10.07	**-4.09**
MuLE-TSR Pattern 46	-2.14	-14.31	0.54	32.36
MuLE-TSB Pattern 23	**-8.20**	**-20.38**	**-10.47**	26.57

Table 6 shows the best BD-Rate results for MuLE-TSR (scanning pattern 46) and MuLE-TSB (scanning pattern 23) as well as for the 4D-Prediction mode (WaSP) with the Verification Model software VM2.1 [51]. MuLE-TSB shows now the best RD performance for three out of the four test LFs; the *Fountain* LF still performs poorly regarding WaSP. As there are clear RD performance improvements (notably for *Fountain*) regarding the MuLE-TSR codec, it may be concluded that the binary-tree-oriented bit-plane clustering tool is more efficient than RLE in the context of these 4D-DCT based LF codecs.

VI. MuLE-MTH: ADOPTING MULTI-SCALE 4D-DCT AND HEXADECA-TREE-ORIENTED BIT-PLANE CLUSTERING

To obtain a thoroughly 4D-native coding solution, the MuLE-TSB coding solution previously proposed may still be further improved. In MuLE-TSB, the coefficient clustering module includes 4D coefficient scanning followed by a binary-tree-oriented bit-plane clustering. However, despite being 4D, the coefficient scanning process generates a 1D array of quantized coefficients that is input to the bit-plane clustering module. To more fully exploit the 4D structure and redundancy among the 4D-DCT coefficients, it is possible to abandon the 4D scanning that generates a 1D array of coefficients to directly perform the 4D-DCT coefficients clustering by replacing the binary-tree-oriented bit-plane clustering by a native 4D, tree-oriented bit-plane clustering; this solution effectively regards the 4D-DCT coefficients as belonging to a 4D structure. This improvement may be achieved by segmenting the 4D block along its four dimensions at each node of the tree. That is, at each node of the tree, the 4D block of transform coefficients is partitioned into 16 4D sub-blocks by splitting each one of its four dimensions t, s, v, u into two segments. This tree clustering of the non-significant 4D transform coefficients and localizing of the significant ones is performed through a so-called *hexadeca-tree*. Fig. 8 illustrates an hexadeca-tree, highlighting a partition of a 4D block into 16 sub-blocks.

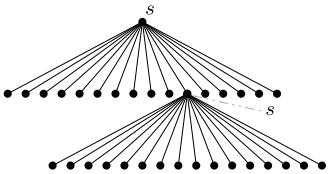


FIGURE 8. Hexadeca-tree structure highlighting a partition of a 4D block into 16 sub-blocks.

Moreover, video codecs such as H.264/AVC [20] and HEVC [21] employ variable-size transforms to better match the transform dimensions to the scale of the block's features, thus improving the compression efficiency; this adaptability has been paramount to increase the video coding compression performance. Therefore, it is natural to attempt improving the 4D-DCT based codec by adopting a variable-size 4D-DCT, which should be able to adapt to the LF data to better exploit its full redundancy. In the LF coding framework presented in Fig. 2, this corresponds to performing the 4D Block Partitioning using variable-sized or multi-scale blocks.

While the partitioning and clustering modules will change, the remaining modules in the framework of Fig. 2, namely the 4D-DCT Transform, Quantization and Entropy Coding are the same as those adopted in the MuLE-TSB codec. This improved LF codec, thoroughly 4D-native, will be referred as **Mu**ltidimensional **L**ight field **E**ncoder using 4D **M**ultiscale **T**ransforms and **H**exadeca-tree-oriented bit-plane Clustering (**MuLE-MTH**). Following the framework in Fig. 2, Subsection VI-A will present the proposed 4D block partitioning

using multi-scale blocks, and Subsection VI-B will detail the proposed coefficients clustering using an hexadeca-tree-oriented bit-plane clustering.

A. 4D BLOCK MULTI-SCALE BLOCK PARTITIONING

Initially, the LF is pre-partitioned into non-overlapping 4D blocks of pre-determined maximum dimensions. Each block thus generated can be further partitioned into a set of non-overlapping 4D sub-blocks according to an RD criterion. The partition process aims at minimizing a Lagrangian coding cost J as defined in Eq. (6). Naturally, the optimal partition has to be signalled, here using a tree structure defined by ternary flags. Each of these flags signals whether:

- The 4D sub-block is not segmented and its 4D-DCT is computed; this corresponds to the flag *noSplit*.
- The 4D sub-block is segmented into four sub-blocks along the (u, v) (spatial) dimensions. This corresponds to the flag *spatialSplit*.
- The 4D sub-block is segmented into four sub-blocks along the (s, t) (view) dimensions; this corresponds to the flag *viewSplit*.

The RD optimized recursive procedure tasked to generate and insert in *codeString* the ternary flags signalling the block partition information is given by Proc. 3 below. Note that this procedure calls the *Recursive hexadeca-tree-oriented bit-plane clustering* procedure specified in Subsection VI-B, which in turn is a straightforward extension to 4D of Proc. 2, and also appends symbols to the same *codeString*, as described in Subsection V-A.

Proc. 3: Recursive optimum 4D block partition

1) *Inputs*:
   - *4D Block*: 4D data block containing the 4D-DCT coefficients to be encoded.
   - m, n, k and l (u, v, s and t) are set to the block dimensions.
   - Minimum sizes for each dimension that still allow a 4D block to be partitioned, namely $M_{\min}$, $N_{\min}$, $K_{\min}$ and $L_{\min}$.
   - Lagrangian multiplier λ to be used in the computation of the coding costs according to Eq. (6).
   - *codeString*: global variable containing the string of symbols to be forwarded to the entropy encoder.
2) *Outputs*:
   - Global variable *codeString* with partition symbols appended.
   - Coding cost, J, for *Block*.
3) *Initialization*:
   - Costs $J_{spatialSplit}$ and $J_{viewSplit}$ are set to infinity.
4) *4D Block partitioning*:
   i. The 4D data block is transformed using a full-size 4D-DCT, its coefficients encoded using the *Recursive hexadeca-tree-oriented bit-plane-clustering* procedure in Subsection VI-B, and the Lagrangian cost returned is attributed to $J_{noSplit}$;
   ii. If $m \leq N_{\min}$ and $n \leq N_{\min}$, then go to to Step 4iv
   iii. The $m \times n \times k \times l$ 4D data block is partitioned into four sub-blocks by splitting it along its spatial dimensions (u, v), generating four 4D sub-blocks of dimensions (as illustrated in Fig. 9a):
      - $\lfloor m/2 \rfloor \times \lfloor n/2 \rfloor \times k \times l$,
      - $(m - \lfloor m/2 \rfloor) \times \lfloor n/2 \rfloor \times k \times l$,
      - $\lfloor m/2 \rfloor \times (n - \lfloor n/2 \rfloor) \times k \times l$ and
      - $(m - \lfloor m/2 \rfloor) \times (n - \lfloor n/2 \rfloor) \times k \times l$.

      Then Proc. 3 is recursively called for each of the four sub-blocks, and the coding costs returned by these four calls are added to compute the cost $J_{spatialSplit}$;
   iv. If $k \leq K_{\min}$ and $l \leq L_{\min}$, then go to to Step 4vi
   v. The $m \times n \times k \times l$ 4D data block is partitioned into four sub-blocks by splitting it along its view dimensions (s, t), generating four 4D sub-blocks of dimensions (as illustrated in Fig. 9b):
      - $m \times n \times \lfloor k/2 \rfloor \times \lfloor l/2 \rfloor$,
      - $m \times n \times (k - \lfloor k/2 \rfloor) \times \lfloor l/2 \rfloor$,
      - $m \times n \times \lfloor k/2 \rfloor \times (l - \lfloor l/2 \rfloor)$,
      - $m \times n \times (k - \lfloor k/2 \rfloor) \times (l - \lfloor l/2 \rfloor)$.

      Then Proc. 3 is recursively called for each of the four sub-blocks, and the coding costs returned by these four calls are added to compute the cost $J_{viewSplit}$;
   vi. If $J_{noSplit} \leq \min\{J_{spatialSplit}, J_{viewSplit}\}$, set the coding cost J to $J_{noSplit}$ and append *noSplit* to *codeString*.
   vii. If $J_{spatialSplit} \leq \min\{J_{noSplit}, J_{viewSplit}\}$, set the coding cost J to $J_{spatialSplit}$ and append *spatialSplit* to *codeString*.
   viii. If $J_{viewSplit} \leq \min\{J_{noSplit}, J_{spatialSplit}\}$, set the coding cost J to $J_{viewSplit}$ and append *viewSplit* to *codeString*.
   ix. The coding cost J is returned and the procedure terminates.

This recursive partition procedure outputs the best partition tree for each initial 4D block with the selected maximum dimensions. Each of its partition nodes signals whether the corresponding 4D sub-block is split along the spatial or

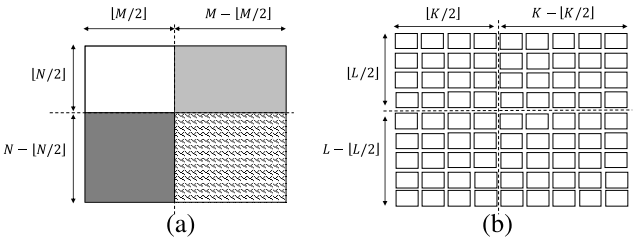


FIGURE 9. (a) Spatial partitioning of a 4D block with spatial dimensions $M \times N$; (b) View partitioning of a 4D block with view dimensions $K \times L$ (adapted from [10]).

view dimensions. Its leaves indicate that the corresponding sub-block is transformed and the bit-planes of its transform coefficients are encoded using hexadeca-tree-oriented bit-plane clustering and an arithmetic encoder. The flags defining the optimum partition tree are also encoded using an arithmetic encoder.

B. HEXADECA-TREE-ORIENTED BIT-PLANE CLUSTERING

In MuLE-MTH, the transform coefficient bit-planes are encoded using RD optimized hexadeca-tree-oriented bit-plane clustering procedure. This procedure is essentially analogous to the one presented in Subsection V-A, with the key difference that each *splitBlock* flag indicates now that a 4D block of transform coefficients is partitioned into 16 4D sub-blocks, as illustrated in Fig. 8. Therefore, the MuLE-MTH procedure *Recursive hexadeca-tree-oriented bit-plane clustering* is very similar to the MuLE-TSB procedure *Recursive binary-tree-oriented bit-plane clustering* defined in detail in Subsection V-A. The main difference is that Proc. 2 should be modified such that the procedure is called recursively for each one of the 16 partitions, and the coding cost of the 16 partitions is added to form the total cost J_s. The partition is such that the dimension of size x, $x = m, n, k, l$, is split into partitions of sizes $\lfloor x/2 \rfloor$ and $x - \lfloor x/2 \rfloor$, where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x.

C. MuLE-MTH RD PERFORMANCE ASSESSMENT

The MuLE-MTH RD performance assessment is made using again the JPEG Pleno CTC presented in Section VII-A and already used for the previous MuLE codecs.

To assess the impact of each of the new multi-scale 4D-DCT and the hexadeca-tree-oriented bit-plane clustering tools on the improved MuLE-MTH RD performance, some comparisons will be performed, notably using a MuLE-MTH abducted version where only fixed transform dimensions are used, labelled as MuLE-TH (dropping the 'M' from multi-scale transform). Thus, by comparing MuLE-MTH and MuLE-TH, it it possible to assess the performance impact of the multi-scale 4D-DCT tool. Moreover, by comparing MuLE-TH with MuLE-TSB and MuLE-TSR, it is possible to assess the performance impact of the successive coding tools integrated into such codecs, excluding the multi-scale 4D-DCT, since the 4D-DCT is exactly the same (with fixed dimensions) for these three MuLE codecs.

Table 7 presents the BD-Rate results for MuLE-MTH in comparison with MuLE-TH for two 4D block dimension configurations, notably $(t, s, v, u) = (13, 13, 31, 25)$ and $(t, s, v, u) = (13, 13, 109, 125)$; this should allow the assessment of the impact of the multi-scale 4D-DCT on the RD performance. For MuLE-MTH, the minimum allowed 4D block dimensions after full 4D block partitioning are set to $(L_{\min}, K_{\min}, N_{\min}, M_{\min}) = (13, 13, 4, 4)$, that is, the 4D blocks are allowed to be partitioned only along the u and v dimensions. The BD-Rate results show that MuLE-MTH with larger 4D blocks outperforms MuLE-MTH with smaller blocks and MuLE-TH with smaller and larger blocks. This

TABLE 7. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-TH and MuLE-MTH (using transform dimensions $(t, s) = (13, 13)$ and (v, u) as specified in second column).

	(v, u)	*Bikes*	*Danger*	*Pillars*	*Fountain*
MuLE-TH	(31,25)	-27.16	-33.31	-25.85	-2.92
MuLE-TH	(109,125)	-22.25	-37.78	-26.92	-10.04
MuLE-MTH	(31,25)	-29.53	-34.48	-26.95	-4.15
MuLE-MTH	(109,125)	**-34.51**	**-43.83**	**-35.79**	**-17.12**

TABLE 8. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-TSR, MuLE-TSB, MuLE-MTH (using transform dimensions $(t, s) = (13, 13)$ and (v, u) as specified in second column) and JPEG Pleno 4D-Prediction mode (WaSP) [51].

	(v, u)	*Bikes*	*Danger*	*Pillars*	*Fountain*
4D-Prediction mode (WaSP)	-	1.43	-13.15	-10.07	-4.09
MuLE-TSR Pattern 46	(31,25)	-2.14	-14.31	0.54	32.36
MuLE-TSB Pattern 23	(31,25)	-8.20	-20.38	-10.47	26.57
MuLE-MTH	(109,125)	**-34.51**	**-43.83**	**-35.79**	**-17.12**

demonstrates that the multi-scale 4D-DCT indeed provides gains in compression efficiency as it allows adapting the block dimensions to the content while starting from larger blocks. The MuLE-TH RD performance is not the best for the larger 4D blocks because MuLE-TH cannot split the 4D blocks to adapt the block size to the specific content within each block as MuLE-MTH does by using the multi-scale 4D-DCT; when using rigid block dimensions, the largest block dimensions are not necessarily the best as the blocks may be obliged to include highly non-redundant data. It is important to remind that all 4D blocks, whatever their dimensions, are independently coded, thus offering random access.

Table 8 shows the BD-Rate values for MuLE-TSR and MuLE-TSB using 4D block dimensions of $(t, s, v, u) = (13, 13, 31, 25)$. By comparing with the MuLE-TH BD-Rate gains for the same 4D block dimensions, it is possible to conclude that MuLE-TH significantly outperforms both MuLE-TSR and MuLE-TSB for all LFs. This clearly indicates that the 4D-native hexadeca-tree-oriented bit-plane clustering is indeed the most effective way for clustering the 4D-DCT coefficients among those investigated. By comparing the BD-Rate gains shown in Tables 7 and 8, it is possible to conclude that the 4D-native hexadeca-tree-oriented bit plane clustering is the most impactful tool on the improved MuLE-MTH RD performance.

Table 8 includes the BD-Rate gains for the best configurations of all MuLE codecs as well as the 4D-Prediction mode (WaSP), always using the JPEG Pleno HEVC anchor [11] as reference. The results show a major MuLE-MTH RD performance improvement, which is now able to outperform not only the HEVC anchor but also all the previous MuLE codecs and the 4D-Prediction mode for all LFs, including the most difficult *Fountain* LF. For the results presented above, while the best MuLE-TSR and MuLE-TSB configurations

use 4D block dimensions of $(t, s, v, u) = (13, 13, 31, 25)$, MuLE-MTH uses a larger block dimension as the block may be dynamically segmented depending on its content. These RD performance results clearly show why MuLE-MTH was accepted for inclusion in the JPEG Pleno Light Field Coding standard as the 4D-Transform Mode [36], [37], [40]. The JPEG Pleno Verification Model software includes the MuLE-MTH codec software and is available at [51].

VII. MuLE-MTH VERSUS LIGHT FIELD CODING STATE-OF-THE-ART Benchmarking

Finally, the most advanced, designed 4D-native LF codec, MuLE-MTH, standardized as 4D-Transform mode in the JPEG Pleno Light Field Coding standard will be compared in terms of RD performance with several state-of-the-art LF codecs. It is important to refer at this stage that, as far as it is known, the MuLE-MTH codec will be a royalty free LF codec, following the JPEG tradition. While this feature is not under the JPEG committee control, this committee has largely advertised its wish to have in each of its standards at least a baseline codec which should be royalty free. In fact, MuLE-MTH corresponds to the Baseline Block-based profile of the JPEG Pleno Light Field Coding standard.

A. EXPERIMENTAL CONDITIONS

To obtain solid comparisons and conclusions in terms of RD performance, it is essential that meaningful experimental conditions are used. In this case, the experimental conditions are, naturally, those specified by the JPEG Pleno Common Test Conditions (CTC) document [11] as they have been driving the most relevant LF coding standardization process.

1) TEST MATERIAL

Since the MuLE codec adopted as JPEG Pleno 4D-Transform coding mode was selected as a standard due to its better RD performance for densely angular sampled LFs, such as lenslet LFs, it is reasonable that this is the type of LF content selected as test material as this is where the reviewed LF codec has to show its great performance regarding the competition. In this context, the selected test material corresponds to the full set of lenslet LFs listed in JPEG Pleno CTC [11], notably *Bikes*, *Danger de Mort* (*Danger* in short), *Stone Pillars Outside* (*Pillars* in short) and *Fountain&Vincent2* (*Fountain* in short) [53], which central SA images are shown in Fig. 10. While these LFs originally included 15×15 PIs with 625×434 resolution, only the 13×13 central PIs are considered for coding due to the intense vignetting effect [11]. The views are made available as 10-bit RGB images in PPM format.

2) CODING CONDITIONS

The selected LFs are coded for the rates, in bits per pixel (bpp), specified in the JPEG Pleno CTC [11], which should be computed as the ratio between the total number of bits used to code the LF and the total number of pixels in the LF. For the lenslet LFs, the recommended rates are 0.001 bpp, 0.005 bpp, 0.02 bpp, 0.1 bpp and 0.75 bpp.

FIGURE 10. Central views for the JPEG Pleno CTC lenslet LFs: ***Bikes*** (top left), ***Danger de Mort*** (top right), ***Stone Pillars Outside*** (bottom left) and ***Fountain&Vincent2*** (bottom right).

3) PERFORMANCE METRICS

The objective quality metric adopted is the PSNR-YUV, in dB, defined as specified in the JPEG Pleno CTC [11]:

$$\text{PSNR-YUV} = \frac{6\,\text{PSNR-Y} + \text{PSNR-U} + \text{PSNR-V}}{8} \quad (9)$$

The PSNR computation assumes a dynamic range of 10 bits, and the input and output LF views must be in RGB format, which is converted to YUV 4:4:4 prior to the PSNR computation [11]. The reported PSNR-YUV values represent the average of the PSNRs for all 13×13 coded views.

4) BENCHMARKS

To exhaustively assess the MuLE-MTH compression efficiency, its RD performance is compared with standard adapted and state-of-the-art LF coding solutions from the literature, notably:

- *HEVC* anchor *(HEVC)*, which corresponds to the pseudo-video like HEVC coding of the 13×13 views in a IPPP serpentine scanning pattern (left-right, top-down) using the x265 [54] implementation [11]; this is the solution taken as reference for all BD-Rate values in this paper.
- *4D-Prediction mode (WaSP)*, which corresponds to the other LF coding mode specified in the JPEG Pleno Light Field Coding standard [10], using the JPEG Pleno VM2.1 Verification Model software [51];
- *HEVC-HR*, which corresponds to the LF codec specified in [23] and reviewed in Section II;
- *MV-HEVC-based*, which corresponds to the LF codec specified in [28] and reviewed in Section II;
- *VVC-serpentine*, which corresponds to the pseudo-video-like VVC coding of the 13×13 views in a IPPP serpentine (left-right, top-down) scanning pattern [11]; while more optimized scannings could be used, serpentine scanning is used here to follow the JPEG Pleno CTC recommendations as for the HEVC anchor. Due to

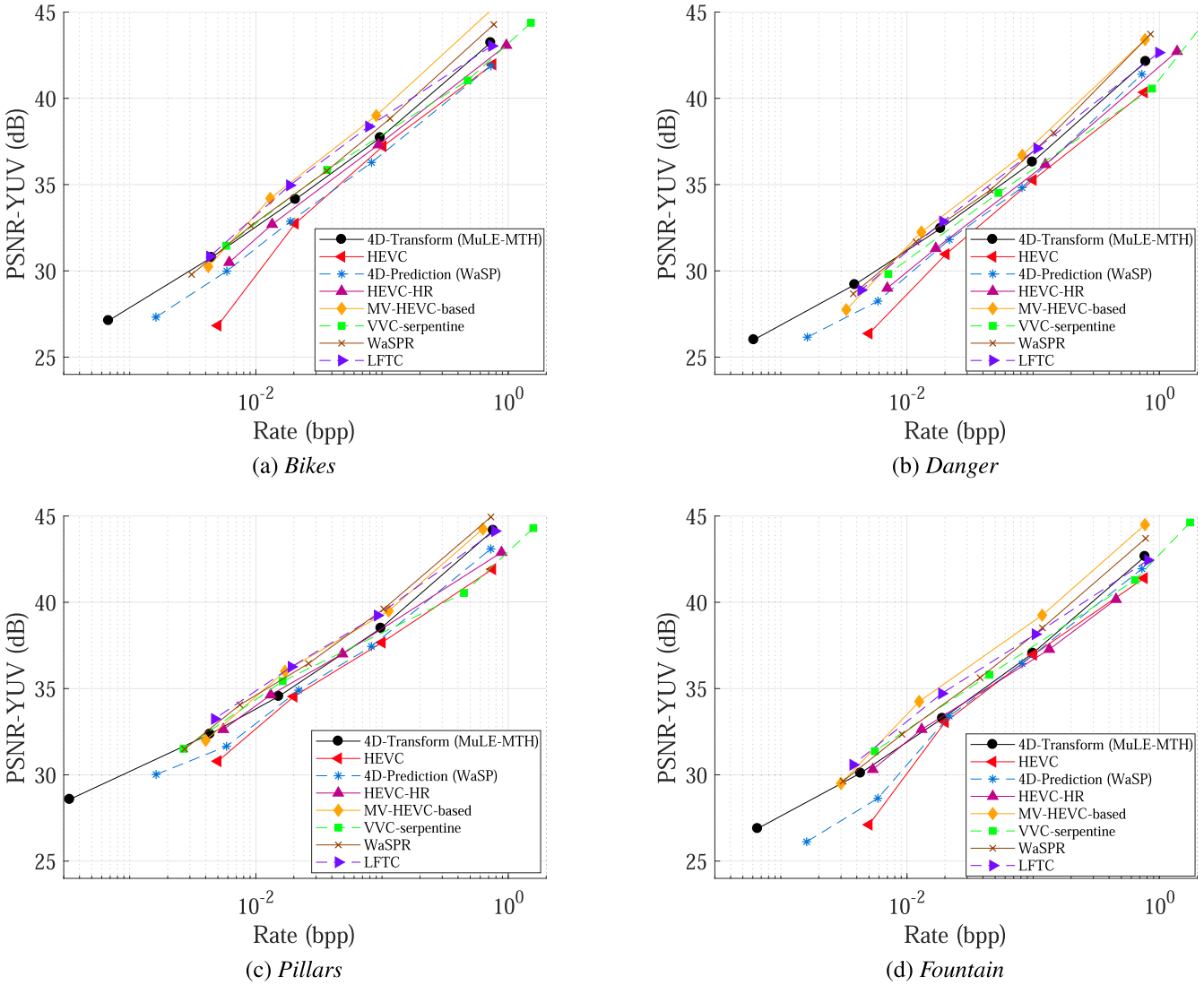


FIGURE 11. RD performance comparison between JPEG Pleno 4D-Transform mode (MuLE-MTH) and benchmarks for the JPEG Pleno CTC lenslet LFs.

constraints of the VVC reference software [55], and to allow a fairer comparison, the LF views were cropped to 624× 432.

- *WaSPR*, which corresponds to the LF codec specified in [38] and reviewed in Section II;
- *LFTC*, which corresponds to the LF codec specified in [39] and reviewed in Section II.

Next sub-section will report and analyze the full set of RD performance results.

B. RD PERFORMANCE ASSESSMENT

Table 9 and Figs. 11a to 11d present the BD-Rates and RD performance for the designed MuLE-MTH (JPEG Pleno 4D-Transform mode, VM2.1) codec, using $(t, s, v, u) = (13, 13, 109, 125)$ as maximum transform dimensions, and the selected benchmarks. The RD performance is expressed as PSNR-YUV (dB) versus rate (bpp).

TABLE 9. BD-Rate (%) regarding the JPEG Pleno HEVC anchor [11] for MuLE-MTH and selected benchmarks.

	Bikes	*Danger*	*Pillars*	*Fountain*
4D-Transform mode (MuLE-MTH)	-34.51	-43.83	-35.79	-17.12
4D-Prediction mode (WaSP) [51]	1.43	-13.15	-10.07	-4.09
HEVC-HR [23]	-20.53	-18.72	-29.11	-6.68
MV-HEVC-based [28]	**-59.31**	**-56.49**	-55.76	**-59.70**
VVC-serpentine [19]	-33.87	-25.00	-34.35	-26.32
WaSPR [38]	-43.31	-48.98	-57.35	-39.99
LFTC [39]	-54.58	-50.90	**-58.75**	-46.91

The obtained RD performance results allow deriving the following conclusions:

- MuLE-MTH (4D-Transform mode) clearly outperforms the JPEG Pleno HEVC anchor and the 4D-Prediction

mode (WaSP) for all JPEG Pleno CTC lenslet LFs, thus fully justifying its inclusion in the JPEG Pleno Light Field Coding standard due to its performance for this type of LFs.

- WaSPR, MV-HEVC-based and LFTC tend to outperform MuLE-MTH for all LFs, especially for the medium to high rates. However, MuLE-MTH is rather competitive regarding VVC offering interesting compression gains, notably for the medium and high rates.
- For the lowest rates, MuLE-MTH shows rate savings when compared to all other codecs except LFTC.
- While MuLE-MTH adopts a rather straightforward 4D-native architecture, notably a transform-quantization-entropy coding pipeline, the benchmarks adopt rather sophisticated prediction and synthesis tools, e.g. intra prediction for HEVC and VVC, and DIBR for WaSPR and LFTC.
- MuLE-MTH does not rely on depth maps or any kind of prediction structure, filters or transforms other than the 4D-DCT. Moreover, it offers random access at block-level what may be a critical feature for some applications.
- The 4D-Prediction mode (WaSP) relies on depth data to efficiently encode information provided by plenoptic cameras, while the 4D-Transform mode is depth agnostic. The availability or not of reliable depth data in a specific application scenario may be a critical factor to decide which JPEG Pleno LF coding mode to use.
- Both the 4D-Transform (MuLE-MTH) and 4D-Prediction (WaSP) modes tend to perform worst than most of the other benchmarks, on average, which rely on very efficient standard coding solutions, burdened by royalties. A key difference between WaSP and WaSPR is precisely the substitution of JPEG 2000 by HEVC. This RD performance penalty is precisely the price of adopting a royalty free approach, which has been dominating the imaging ecosystem for the past decades (differently from video).

In summary, the RD performance results for MuLE-MTH show that this is a very competitive LF coding solution for densely angular sampled LFs, notably if royalty free licensing is also a goal. Finally, it is worthwhile to note that the MuLE-MTH decoder has low computational complexity, since it just has to decode the block partition tree and the hexadeca-tree, which are rather simple, and thus very fast operations. On the other hand, the encoder computational complexity, which is largely associated to the recursive 4D block partitioning described in Proc. 3, can be traded off with RD performance. This computational complexity strongly depends on the block dimensions and the minimum sizes for each dimension that still allow a 4D block to be partitioned. The larger the block dimensions and the smaller its minimum values, the better is the RD performance but the higher is the associated computational complexity. However, it is relevant to observe that good RD performance trade-offs may be obtained while keeping the computational complexity within acceptable bounds, not forgetting that the computational complexity is less critical for still images than for video.

VIII. CONCLUSIONS AND FUTURE WORK

This paper reports the design and integration processes which led to the first 4D-native LF codecs based on the 4D-DCT, following a sequence of innovation steps. The initial LF codec (MuLE-TSR) is a straightforward 4D extension of the JPEG Baseline image coding standard, basically employing a 4D transform, quantization, 4D coefficient scanning, run-length encoding (RLE) and arithmetic coding. After, its RD performance has been improved by replacing the RLE by a binary-tree-oriented bit-plane clustering of the 4D-DCT coefficients (MuLE-TSB). Finally, an end-to-end 4D-native codec has been obtained by replacing the binary-tree-oriented bit-plane clustering by a 4D-native hexadeca-tree-oriented bit-plane clustering and adopting an RD-based control of the 4D block dimensions (MuLE-MTH). MuLE-MTH shows competitive RD performance when compared to more sophisticated LF coding solutions, with the advantage of being conceptually simple and royalty-free, not relying on depth maps and offering block-based random access. Due to these advantages, MuLE-MTH has been selected as one of the two coding modes specified in the JPEG Pleno Light Field Coding standard, specifically targeting densely angular sampled LFs such as lenslet LFs [36], [37], [40].

While offering competitive RD performance, the truth is that the MuLE-MTH LF codec might still be further improved. In this context, two research directions are here presented as future work. The first is the introduction of intra prediction tools between the 4D blocks, which should allow to exploit the 4D redundancy also across 4D blocks and not just within 4D blocks by coding the predicted 4D blocks with a lower energy block residue. Another key research direction targets improving the MuLE-MTH RD performance for less densely angular sampled LFs where the view disparity is larger. This target may be accomplished by replacing the orthogonal 4D-DCT by a 4D transform slanted along the edges of the epipolar plane images, with the objective of increasing the energy compaction of the 4D transform coefficients.

At this stage, it is also time to check how the LF devices and applications market will evolve, with the first big signs coming from the mobile world, which devices do not stop increasing the number of cameras included, in practice building richer LFs.

ACKNOWLEDGMENT

The authors would like to thank the authors of [23], [38], [39] for sharing their results and the authors of [19] for providing their configuration files.

REFERENCES

[1] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, F. M. Landy and J. A. Movshon, Eds. Cambridge, MA: MIT Press, 1991, pp. 3–20.

[2] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, New York, NY, USA, 1996, pp. 31–42.

[3] A. Kondoz and T. Dagiuklas, *Novel 3D Media Technologies*. New York, NY, USA: Springer-Verlag, 2015.

[4] R. Ng, "Digital light field photography," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Jul. 2006. [Online]. Available: http://www.lytro.com/renng-thesis.pdf

[5] *JPEG Pleno Abstract and Executive Summary*, document ISO/IEC JTC 1/SC 29/WG1 N69022, 69th JPEG Meeting, Sydney, NSW, Australia, 2015. [Online]. Available: https://jpeg.org/jpegpleno/documentation.html

[6] *JPEG Pleno Scope, Use Cases and Requirements*, document ISO/IEC JTC 1/SC 29/WG1 N74020, 74th JPEG Meeting, Geneva, Switzerland, Jan. 2017. [Online]. Available: https://jpeg.org/jpegpleno/documentation.html

[7] *JPEG Pleno Call for Proposals on Light Field Coding*, document ISO/IEC JTC 1/SC 29/WG1 N74014, 74th JPEG Meeting, Geneva, Switzerland., Jan. 2017. [Online]. Available: https://jpeg.org/jpegpleno/documentation.html

[8] *Final Call for Evidence on JPEG Pleno Point Cloud Coding*, document ISO/IEC JTC 1/SC 29/WG1 N88014, 88th JPEG Meeting, Jul. 2020. [Online]. Available: https://jpeg.org/jpegpleno/documentation.html

[9] *Draft Call for Proposals on JPEG Pleno Holography*, document ISO/IEC JTC 1/SC 29/WG1 N88016, 88th JPEG Meeting, Jul. 2020. [Online]. Available: https://jpeg.org/jpegpleno/documentation.html

[10] *JPEG Pleno Part 2–ISO/IEC FDIS 21794-2*, document ISO/IEC JTC1/SC29/WG1, WG1N87033, 87th JPEG Meeting, Erlangen, Germany, Apr. 2020.

[11] F. Pereira, C. Pagliari, E. A. B. da Silva, I. Tabus, H. Amirpour, M. Bernardo, and A. Pinheiro, *Information technology - JPEG Pleno Light Field Coding Common Test Conditions V3.3*, document ISO/IEC JTC 1/SC29/WG1N84025, 84th JPEG Meeting, Brussels, Belgium, 2019.

[12] C. Brites, J. Ascenso, and F. Pereira, "Lenslet light field image coding: Classifying, reviewing and evaluating," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Feb. 27, 2020.

[13] C. Conti, L. D. Soares, and P. Nunes, "Dense light field coding: A survey," *IEEE Access*, vol. 8, pp. 49244–49284, 2020.

[14] S.-H. Tsang, Y.-L. Chan, and W. Kuang, "Standard-compliant HEVC screen content coding for raw light field image coding," in *Proc. 13th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2019, pp. 1–6.

[15] S.-H. Tsang, Y.-L. Chan, and W. Kuang, "Mode skipping for HEVC screen content coding via random forest," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2433–2446, Oct. 2019.

[16] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 50–62, Jan. 2016.

[17] S.-H. Tsang, Y.-L. Chan, W. Kuang, and W.-C. Siu, "Reduced-complexity intra block copy (IntraBC) mode with early CU splitting and pruning for HEVC screen content coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 269–283, Feb. 2019.

[18] R. Monteiro, L. Lucas, C. Conti, P. Nunes, N. Rodrigues, S. Faria, C. Pagliari, E. da Silva, and L. Soares, "Light field HEVC-based image coding using locally linear embedding and self-similarity compensated prediction," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2016, pp. 1–4.

[19] V. Avramelos, J. D. Praeter, G. Van Wallendael, and P. Lambert, "Light field image compression using versatile video coding," in *Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Sep. 2019, pp. 1–6.

[20] *Advanced Video Coding for Generic Audiovisual Services, Information Technology-Coding of Audio-Visual Objects–Part 10: Advanced Video Coding (MPEG-4 AVC)*, document ITU-T H.264 and ISO/IEC 14496-10, 2014.

[21] *High Efficiency Video Coding*, document ITU-T H.265 and ISO/IEC 23008-2, 2013.

[22] *Working Draft 4 of Versatile Video Coding Joint Video Experts Team (JVET)*, document ITU-T SG 16 WP3 and ISO/IEC JTC 1/SC29/WG11 N18274, 13th Meeting, Marrakech, Morocco, Jan. 2019.

[23] R. J. S. Monteiro, N. M. M. Rodrigues, S. M. M. Faria, and P. J. L. Nunes, "Light field image coding based on hybrid data representation," *IEEE Access*, vol. 8, pp. 115728–115744, 2020.

[24] W. Tu, X. Jin, L. Li, C. Yan, Y. Sun, M. Xiao, W. Han, and J. Zhang, "Efficient content adaptive plenoptic video coding," *IEEE Access*, vol. 8, pp. 5797–5804, 2020.

[25] H. Han, J. Xin, and Q. Dai, "Plenoptic image compression via simplified subaperture projection," in *Advances in Multimedia Information Processing (PCM)*. Cham, Switzerland: Springer, Sep. 2018, pp. 274–284.

[26] W. Ahmad, S. Vagharshakyan, M. Sjostrom, A. Gotchev, R. Bregovic, and R. Olsson, "Shearlet transform-based light field compression under low bitrates," *IEEE Trans. Image Process.*, vol. 29, pp. 4269–4280, 2020.

[27] J. Gu, B. Guo, and J. Wen, "High efficiency light field compression via virtual reference and hierarchical MV-HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 344–349.

[28] W. Ahmad, M. Ghafoor, S. A. Tariq, A. Hassan, M. Sjostrom, and R. Olsson, "Computationally efficient light field image compression using a multiview HEVC framework," *IEEE Access*, vol. 7, pp. 143002–143014, 2019.

[29] L. Wei, Y. Wang, and Y. Liu, "Tensor-based light field compressed sensing and epipolar plane images reconstruction via deep learning," *IEEE Access*, vol. 8, pp. 134898–134910, 2020.

[30] M. P. Pereira, G. Alves, C. L. Pagliari, M. B. de Carvalho, E. A. B. da Silva, and F. Pereira, "A geometric space-view redundancy descriptor for light fields: Predicting the compression potential of the JPEG pleno light field datasets," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2017, pp. 1–6.

[31] G. Alves, M. P. Pereira, M. B. de Carvalho, F. Pereira, C. L. Pagliari, V. Testoni, and E. A. B. da Silva, "A study on the 4D sparsity of JPEG pleno light fields using the discrete cosine transform," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 1148–1152.

[32] M. B. de Carvalho, M. P. Pereira, G. Alves, E. A. B. da Silva, C. L. Pagliari, F. Pereira, and V. Testoni, "A 4D DCT-based lenslet light field codec," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 435–439.

[33] T. Zhong, X. Jin, L. Li, and Q. Dai, "Light field image compression using depth-based CNN in intra prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 8564–8567.

[34] E. Dib, M. L. Pendu, and C. Guillemot, "Light field compression using Fourier disparity layers," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 3751–3755.

[35] P. Astola and I. Tabus, "WaSP: Hierarchical warping, merging, and sparse prediction for light field image compression," in *Proc. 7th Eur. Workshop Vis. Inf. Process. (EUVIP)*, Nov. 2018, pp. 1–6.

[36] P. Schelkens, P. Astola, E. A. B. da Silva, C. Pagliari, C. Perra, I. Tabus, and O. Watanabe, "JPEG Pleno light field coding technologies," *Proc. SPIE*, vol. 11137, pp. 391–401, Sep. 2019.

[37] P. Astola, L. A. da Silva Cruz, E. A. B. da Silva, T. Ebrahimi, P. G. Freitas, A. Gilles, K.-J. Oh, C. Pagliari, F. Pereira, C. Perra, S. Perry, A. M. G. Pinheiro, P. Schelkens, I. Seidel, and I. Tabus, "JPEG Pleno: Standardizing a coding framework and tools for plenoptic imaging modalities," *ITU J., ICT Discoveries*, vol. 3, no. 1, pp. 1–15, Jun. 2020. [Online]. Available: http://handle.itu.int/11.1002/pub/8153d79a-en

[38] P. Astola and I. Tabus, "Coding of light fields using disparity-based sparse prediction," *IEEE Access*, vol. 7, pp. 176820–176837, 2019.

[39] B. Heriard-Dubreuil, I. Viola, and T. Ebrahimi, "Light field compression using translation-assisted view estimation," in *Proc. Picture Coding Symp. (PCS)*, Nov. 2019, pp. 1–5.

[40] C. Perra, P. Astola, E. A. B. da Silva, H. Khanmohammad, C. Pagliari, P. Schelkens, and I. Tabus, "Performance analysis of JPEG Pleno light field coding," *Proc. SPIE*, vol. 11137, pp. 402–413, Sep. 2019.

[41] *JPEG Pleno Part 1–ISO/IEC FDIS 21794-1*, document ISO/IEC JTC1/SC29/WG1, Jan. 2020, WG1N86056, 86th JPEG Meeting, Sydney, NSW, Australia.

[42] *Digital Compression and Coding of Continuous-Tone Still Images–CCIT Recommendation T.81*, document ISO/IEC 10918-1, 1993.

[43] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[44] G. Hudson, A. Léger, B. Niss, I. Sebestyén, and J. Vaaben, "JPEG-1 standard 25 years: Past, present, and future reasons for a success," *J. Electron. Imag.*, vol. 27, no. 4, pp. 1–19, 2018.

[45] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[46] E. Y. Lam and J. W. Goodman, "A mathematical analysis of the DCT coefficient distributions for images," *IEEE Trans. Image Process.*, vol. 9, no. 10, pp. 1661–1666, Oct. 2000.

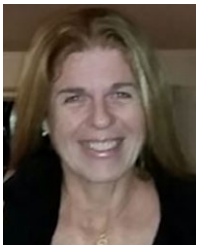
[47] T. Fryza, "Scan order and Huffman coding of 3D DCT coefficientss," in *Proc. 7th WSEAS Int. Conf. Math. Methods Comput. Techn. Electr. Eng.*, 2005, pp. 235–238.

[48] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Upper Saddle River, NJ, USA: Prentice-Hall., 1990.

[49] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.
[50] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T VCEG-M33, 2001. [Online]. Available: https://ci.nii.ac.jp/naid/10016799583/en/
[51] *JPEG Pleno Light Field Verification Model 2.1*. Accessed: Nov. 26, 2019. [Online]. Available: https://gitlab.com/wg1/jpeg-pleno-vm
[52] J. Andrew, "A simple and efficient hierarchical image coder," in *Proc. Int. Conf. Image Process.*, 1997, pp. 658–661.
[53] *JPEG Pleno Light Field Datasets According to Common Test Conditions*. Accessed: Nov. 26, 2019. [Online]. Available: https://jpeg.org/plenodb/lf/pleno_lf/
[54] *x265 HEVC Encoder*. Accessed: Nov. 26, 2019. [Online]. Available: http://x265.org
[55] *VVC Test Model (VTM)*. Accessed: Nov. 26, 2019. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

GUSTAVO DE OLIVEIRA ALVES (Member, IEEE) was born in Cachoeiro de Itapemirim, Brazil. He received the degree in electronics engineering from the Universidade Federal do Rio de Janeiro (UFRJ), Brazil, in 2014, and the M.Sc. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), in 2019, where he is currently pursuing the D.Sc. degree in electrical engineering. From 2011 to 2018, he was a TV Systems Researcher with TV Globo, Rio de Janeiro, Brazil. Since 2019, he has been a Digital TV Engineer with Globoplay, Rio de Janeiro. His research interests include digital television and digital signal/image/video processing.

MURILO BRESCIANI DE CARVALHO (Member, IEEE) was born in Petrópolis, Brazil, in 1964. He received the B.E. degree in electrical engineering from Universidade Federal Fluminense (UFF), Brazil, in 1986, the M.Sc. degree in electrical engineering (telecommunications systems) from the Pontifical Universidade Católica do Rio de Janeiro (PUC-RJ), in 1994, and the D.Sc. degree in electrical engineering (signal processing) from the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), in 2001. Since 1994, he has been with the Department of Telecommunications Engineering, UFF. His research interests include digital image/video processing, source/channel coding, and digital signal processing.



CARLA L. PAGLIARI (Senior Member, IEEE) received the Ph.D. degree in electronic systems engineering from the University of Essex, U.K., in 2000. In 1983 and 1985, she was with TV Globo, Rio de Janeiro, Brazil. From 1986 to 1992, she was a Researcher with the Instituto de Pesquisa e Desenvolvimento, Rio de Janeiro. Since 1993, she has been a Senior Researcher and a Professor with the Department of Electrical Engineering, Military Institute of Engineering (IME), Rio de Janeiro. Her research interests include light field signal processing and coding, image and video processing, and computer vision. She also serves as an Expert on the JPEG Pleno Committee, since January 2018. She is currently a member of the ISO/IEC JTC1/SC29/WG1 (JPEG) standardization committee, the Brazilian delegation for the WG1 JPEG activities, and the Board of Teaching of the Brazilian Society of Television Engineering.

PEDRO GARCIA FREITAS received the B.S. degree in physics from the University of Brasília (UnB), Brazil, in 2010, and the M.S. and Ph.D. degrees in computer science from UnB, in 2010 and 2017, respectively. Prior to his Ph.D., he worked for more than 12 years as a Software Developer in multi-disciplinary projects in the areas of data science, smart grids, information sciences, image processing, and so on. He is currently a Researcher with the Samsung R&D Institute Brazil. His research interests include computer vision, machine learning, and quality of experience.

ISMAEL SEIDEL (Member, IEEE) was born in São Bento do Sul, Santa Catarina, Brazil, in 1990. He received the B.S., M.S., and Ph.D. degrees in computer science from the Federal University of Santa Catarina (UFSC), Florianópolis, Brazil, in 2011, 2014, and 2019, respectively. From 2009 to 2019, he was a Research Assistant with the Embedded Computing Laboratory (ECL), UFSC. Since 2019, he has been a Research Scientist with the Samsung R&D Institute Brazil (SRBR), Campinas, Brazil. His research interests include visual signal processing/coding and related low-power hardware architectures.

MARCIO PINTO PEREIRA (Member, IEEE) received the degree in electronics engineering in 1999, and the M.Sc. degree in 2010. He is currently pursuing the D.Sc. degree in signal processing, electrical engineering with the Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil. He acted as the Director of the Brazilian TV Engineering Society (SET), from 2008 to 2016, and as the Technology Director of Globo TV network affiliates and Globo's educational channel Futura, Brazil, from 2000 to 2015. His research interests include signal processing, image compression, digital TV, and light fields.

CARLA FLORENTINO SCHUELER VIEIRA was born in Niterói, Brazil, in 1993. She received the degree in telecommunications engineering from Universidade Federal Fluminense (UFF), Brazil, in 2018. She is currently pursuing the M.S. degree in electrical engineering with the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Brazil. Her research interest includes signal processing with a focus on image processing.

VANESSA TESTONI (Senior Member, IEEE) received the B.S. degree in computer science from the Pontifical Catholic University of Paraná (PUCPR), Brazil, in 2002, the B.S. degree in electrical engineer from the Federal University of Paraná (UFPR), Brazil, in 2004, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Campinas (UNICAMP), Brazil, in 2006 and 2011, respectively. Right after her graduation, she joined the University of California San Diego (UCSD), USA, as a Postdoctoral Employee. Since 2013, she has been the Research Leader of media standards with SRBR (Samsung Research Institute Brazil), where her research interests include digital signal processing, computer vision, machine learning, and specially image and video coding. She has authored/coauthored several scientific publications and patents. She was the Brazilian Ph.D. Student awarded the Microsoft Research Ph.D. Fellowship Award, in 2009. In 2014, she received the MIT TR35 (Young Innovators under 35) in the First Brazilian Edition of the Award. In 2016, she was elected an Affiliate Member of the Brazilian Academy of Sciences (ABC) and also the Chair of the IEEE SPS (Signal Processing Society) São Paulo Chapter. In 2018, she was nominated the First National Head of the ISO/IEC JTC 001/SC 29 (JPEG/MPEG) Brazilian Delegation.

FERNANDO PEREIRA (Fellow, IEEE) is currently a Professor with the Department of Electrical and Computers Engineering, Instituto Superior Técnico, Universidade de Lisboa, and a Senior Researcher with the Instituto de Telecomunicações, Lisbon, Portugal. He has contributed more than 300 articles in international journals, conferences, and workshops, and made several tens of invited talks at conferences and workshops. His research interests include visual data analysis, coding, description, adaptation, quality assessment, and advanced multimedia services. He is or has been a member of the IEEE Signal Processing Society Technical Committees on Image, Video and Multidimensional Signal Processing, and Multimedia Signal Processing, and the IEEE Circuits and Systems Society Technical Committees on Visual Signal Processing and Communications, and Multimedia Systems and Applications. He was an IEEE Distinguished Lecturer, in 2005, and elected as an IEEE Fellow, in 2008, for contributions to object-based digital video representation technologies and standards. He has been elected to serve on the Signal Processing Society Board of Governors in the capacity of Member-at-Large for a 2012 and a 2014–2016 term. Since January 2018, he has been the SPS Vice-President of Conferences. Since 2013, he has also been a EURASIP Fellow for contributions to digital video representation technologies and standards. He has been elected to serve on the European Signal Processing Society Board of Directors for a 2015–2018 term. Since 2015, he has also been an IET Fellow. He is/has been a member of the Scientific and Program Committees of many international conferences and workshops. He has been the General Chair of the Picture Coding Symposium (PCS) in 2007, the Technical Program Co-Chair of the International Conference on Image Processing (ICIP) in 2010 and 2016, the Technical Program Chair of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS) in 2008 and 2012, and the General Chair of the International Conference on Quality of Multimedia Experience (QoMEX) in 2016. He has been participating in the MPEG standardization activities, notably as the Head of the Portuguese delegation, MPEG Requirements Group Chair, and the Chair of many Ad Hoc Groups related to the MPEG-4 and MPEG-7 standards. Since February 2016, he has also been the JPEG Requirements Group Chair. He is also an Area Editor of the *Signal Processing: Image Communication* journal and an Associate Editor of the *EURASIP Journal on Image and Video Processing*. He is or has been a member of the Editorial Board of the *IEEE Signal Processing Magazine* and an Associate Editor of the IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the *IEEE Signal Processing Magazine*. From 2013 to 2015, he was the Editor-in-Chief of the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING.

EDUARDO A. B. DA SILVA (Senior Member, IEEE) was born in Rio de Janeiro, Brazil. He received the degree in electronics engineering from the Instituto Militar de Engenharia (IME), Brazil, in 1984, the M.Sc. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), in 1990, and the Ph.D. degree in electronics from the University of Essex, U.K., in 1995. He has been a Professor with the Universidade Federal do Rio de Janeiro since 1989. He is the coauthor of the book *Digital Signal Processing: System Analysis and Design* (Cambridge University Press in 2002) that has also been translated to the Portuguese and Chinese languages, whose second edition has been published in 2010. He has published more than 70 articles in international journals. His research interests include signal and image processing, signal compression, digital TV, 3D videos, computer vision, light fields, and machine learning, together with its applications to telecommunications and the oil and gas industry. He is a Senior Member of the Brazilian Telecommunications Society (SBrT). He is the Co-Editor of the future standard ISO/IEC 21794-2 and JPEG Pleno Plenoptic image coding system.