IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 26, 2021, accepted February 12, 2021, date of publication February 16, 2021, date of current version February 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3059921

Lossless Compression of Plenoptic Camera Sensor Images

IOAN TABUS, (Senior Member, IEEE), AND EMANUELE PALMA, (Student Member, IEEE)
Computing Sciences Department, Tampere University, FI-33101 Tampere, Finland

Corresponding author: Ioan Tabus (ioan.tabus@tuni.fi)

The work of Emanuele Palma was funded by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie grant agreement Nr. 764951, Immersive Visual Technologies for Safety-Critical Applications.

ABSTRACT We propose a codec for the lossless compression of plenoptic camera sensor images. The proposed encoder starts by splitting the input lenslet image into rectangular patches, with each patch corresponding to a microlens image. The encoder and decoder exploit the correlation between the pixels in neighbor patches using a patch-by-patch prediction mechanism where each pixel of a patch has its own dedicated sparse predictor designed to utilize the most relevant pixels from the neighbor patch to the left. An intra-patch prediction mask together with the pixels from the neighbor left patch form the final prediction template. The encoder performs the design of sparse predictors by first finding the relevant regressors in the final template. The patches are classified into M classes according to two possible mechanisms (either based on depth information or based on Bayer mask colors), and the sparse predictor design is performed for each pair (class label; patch pixel index). A relevant context selection mirrors the selection of relevant regressors, thereby providing the arithmetic coding with skewed coding distributions at each context. We show examples of the application of the proposed methods on sensor images from the JPEG Pleno database, thus demonstrating the improved performance of the proposed methods compared to the existing predictive methods for encoding camera sensor images.

INDEX TERMS Plenoptic camera, sensor image lossless compression, Bayer sensor lossless compression.

I. INTRODUCTION

Light field (LF) compression is a mature research area that has been extensively discussed in the engineering literature, see e.g., [1] including more than 250 references. Several techniques are already in the process of standardization under the JPEG Pleno standards [2], [3] and under immersive video coding MPEG-I standards [4].

The subfield of *lossless* compression for plenoptic camera images has also been explored in more than dozen of publications in recent years, see [5]–[19]. The grand challenges that were organized on light field compression at ICME 2016 [23] and at ICIP 2017 (jointly with the JPEG Pleno Light Field Call for Proposals [24]) had a strong contribution in promoting a unified test material, containing mainly plenoptic camera datasets extracted from the EPFL Light Field database [25], and in promoting a unified evaluation procedure, particularly for lossy compression [26]. The submissions to ICIP 2017 grand challenge and JPEG Pleno CfP [24] were also requested to provide lossless or near-lossless results.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei-Wen Hu.

A different approach to plenoptic camera image compression was pursued in [5], [8], where the *plenoptic sensor image* was selected to be encoded and decoded, leaving to the decoder the additional task of running the entire plenoptic processing pipeline from the sensor image up to the LF array of views. We follow herein the same approach as that in [5], [8] and present an efficient codec for the plenoptic camera sensor image, called the sparse relevant regressors and contexts (SRRC) codec. We emphasize the connection of this approach with the lossless compression of Bayer pattern images (or color filter array (CFA) images) and to the *compress-first* workflows [30]. Our proposal generalizes the important field of traditional camera sensor image compression to the field of *plenoptic* camera sensor image lossless compression.

A. THE SRRC CODEC AND RELATED WORK ON LOSSLESS COMPRESSION OF BAYER PATTERN IMAGES

The SRRC encoder presented in Section II is related to a long line of research studies on lossless compression of Color Filter Array (CFA) images, also called Bayer pattern images.

We note that the plenoptic camera records data on an image sensor that has in front of it a Bayer pattern mask, also called a color filter array, which has in its turn an array of microlenses in front of it.

The techniques for the lossless compression of CFA images were developed during the last two decades and this research topic is still actively pursued, see the recent papers [31]–[34] and the references therein.

In a summary of the major contributions of the area, [33] cites four methods as important techniques from the point of view of lossless compression performance: lossless compression of mosaic images (LCMI) based on a Mallat packet decomposition [35], lossless compression of real CFA data [36], hierarchical prediction and context modeling (referred to by us here as HPCM) [37], and context matching based prediction (CMBP) [38], grouped into two main families of methods. Although the family of methods based on the subband or wavelet coding methods produced efficient lossless codecs [33], [35], the best reported results are achieved by the second family of methods that are based on predictive encoding, [36], [37], and [38], where the predictor of a pixel, and its context (possibly taking into account the gradients around the pixel to be predicted) are specified differently for the pixels with different Bayer mask color labels.

Compression of CFA images was recently of interest also in the standardization field. The compression of Bayer pattern images was considered in [39] that proposed a simple solution for enhancing the efficiency of JPEG-XS applied directly on Bayer pattern images.

As a motivation of the *lossless* compression of CFA images, most of the cited papers stressed the advantage of first encoding the CFA image in a lossless manner and then having it available for applying various debayering or post-processing algorithms with different settings. Thus, the scene-dependent information is stored in a lossless manner which is an important requirement when archiving valuable images in either professional imaging applications or in medical applications. For example, such an application was recently addressed in [40] that proposes fast versions of lossless compression of CFA images for endoscopy images.

In this paper, we consider plenoptic camera sensor images and introduce predictive techniques for exploiting the correlations in the recorded image due to the array of microlenses (that were also exploited, in some different forms in [5] and [8]). Additionally, we also consider the regularities and types of redundancies specific to Bayer pattern images in a first encoding scheme dubbed SRRC-PHASE, where we take into account the Bayer mask color labels as follows: our predictors are specific for each Bayer mask color label, but additionally, they depend on the alignment with respect to the Bayer mask between the current patch and its neighbor patch (which is used for predicting the current pixel). Thus, in SRRC-PHASE, the specific design of the predictors is intended to utilize both the redundancy due the microlens structure and the redundancy due to the Bayer mask structure of the recorded image.

The second proposed method, named SRRC-DEPTH, uses the same patch-by-patch prediction approach, but with a specific set of sparse predictors designed for the set of pixels situated at each depth level in the scene, capturing different dependencies between the pixels in neighboring microlenses, specific to each depth level.

B. ORGANIZATION OF THE PAPER

In Section II, we present a detailed description of the SRRC codec for the plenoptic camera sensor image, introducing the patch-by-patch coding based on a predictive approach, where both the prediction and the context selection make use of the relevant pixels found in an overall prediction template. Section III exemplifies experimental results and compares them with the results obtained by predictive methods for the lossless compression of CFA images.

II. THE LOSSLESS CODEC SPARSE RELEVANT REGRESSORS AND CONTEXTS (SRRC) FOR PLENOPTIC CAMERA SENSOR IMAGES

In this section, we introduce the encoding algorithm for a sensor image that is dubbed here *Sparse Relevant Regressors and Contexts* (SRRC).

The sensor image is formed at the $N_r \times N_c$ rectangular grid of pixels, by recording the light passing through the $M_r \times M_c$ array of microlenses with each microlens centered at a point of a hexagonal lattice. The lattice has one of its axes roughly aligned with the horizontal direction of the pixels in the sensor. At the same time, the sensor image is a Bayer pattern image because the sensor has a Bayer mask located in front of it.

In Fig. 1, an overall diagram scheme of the proposed encoder is given. The data recorded at the large sensor image that capture the light passing through the array of approximately a quarter of million microlenses are highly redundant. The encoder exploits the redundancy in the data by trying to correlate the data captured under one microlens with the data captured at the neighboring microlens (left neighbor in our implementation).

We decide to encode the sensor image in a patch-by-patch processing, where each patch corresponds approximately to a microlens and is obtained by cropping a rectangular region centered at the sensor grid point that is the closest to an estimated microlens center. One reason for this type of operation is because the same type of rectangular patch cropping is used in the stage of lenslet resampling and rectification, when the final array of views is obtained from the sensor image [41].

To formally define the patches in the overall sensor image, we denote by $\mathbf{X}$ the 5368×7728 sensor image (also called lenslet image) associated with a rectangular grid and we denote the pixel indices as (I, J), with $1 \leq I \leq M_r = 5368$ and $1 \leq J \leq M_c = 7728$.

Each microlens has its center located at a point of a hexagonal lattice. The lattice alignment with respect to the image sensor grid is specific to each dataset. The alignment is specified by five real-valued parameters that we transmit

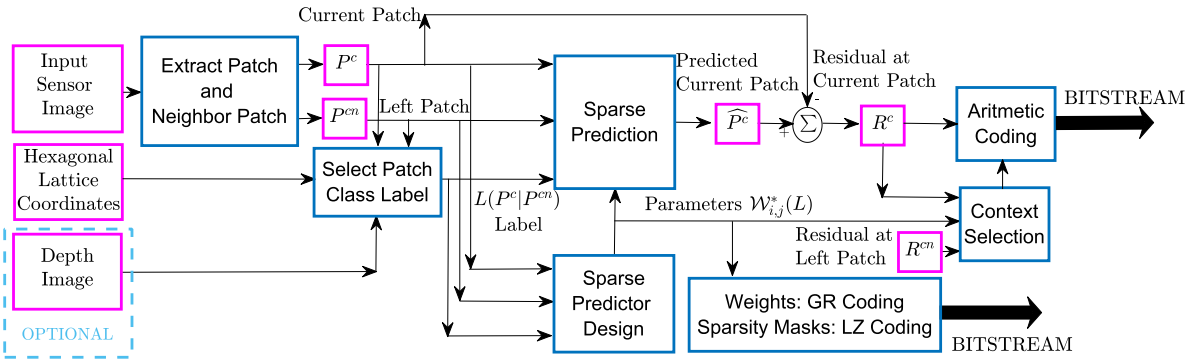

FIGURE 1. Diagram of the sparse relevant regressors and contexts (SRRC) encoder for encoding one camera sensor image.

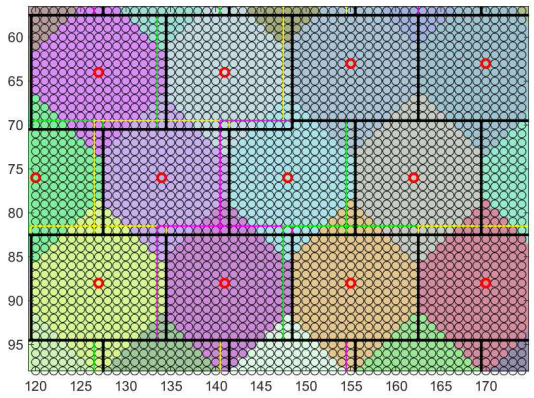

FIGURE 2. Covering the sensor image with rectangular patches (each patch is 13 × 15, shown with thin colored lines) centered at the nodes of the hexagonal lattice. The patches are seen to slightly overlap. Due to the patch-by-patch scanning order and due to the slight overlapping, the pixels processed at each patch are only those enclosed by the thick black lines. The hexagonal Voronoï cells of the hexagonal lattice points are shown shaded in pseudocolor (each Voronoï cell approximately circumscribes a circular microlens). Each rectangular patch contains at each corner approximately 6 unreliable pixels (that propagate after pipeline processing to generate the well-known glitch of unreliable views at the corners of the LF array of views). The centers of the patches (shown as red circles) are located at the various colors of the Bayer pattern [42], because in order to obtain the same Bayer color, two neighbor centers should be separated both horizontally and vertically by an odd number of pixels, which is not the case for every pair of the neighbor centers in the figure.

to the decoder in the full precision form. From these parameters, the ideal positions of the centers of all microlenses that cover the 5368 × 7728 rectangular grid can be found, and these positions are truncated to integers to create integer coordinates for each center of a microlens, as described in [41]. As presented in detail in Fig. 2, we show the pixels from $\mathbf{X}$ that are centers of microlenses that are recovered from the five parameter metadata by red circles. We can cover the sensor image by using only 434×541 microlenses, and each microlens is referred to by a pair (u, v) with $1 \leq u \leq N_r = 434$ and $1 \leq v \leq N_c = 541$. The coordinates in the image $\mathbf{X}$ of the center of the microlens (u, v) are specified by two 434×541 matrices, i.e., the center of the microlens (u, v) has coordinates $(I^0(u, v), J^0(u, v))$ in the image $\mathbf{X}$.

These two matrices are constructed using as necessary information only five parameters (which we save into the encoded archive), specifying geometrical parameters of the hexagonal lattice, and its alignment with respect to the sensor, and hence the two matrices are also available to the decoder. The functions necessary for transforming the five parameters to the set of microlens centers from which we compute $\mathbf{I^0}$ and $\mathbf{J^0}$ are given in the light field MATLAB toolbox [41].

We consider rectangular patches of 13 × 15 pixels, each centered on the center of a microlens (with coordinates given in $\mathbf{I^0}$ and $\mathbf{J^0}$), and we index the patches similarly to the microlenses, i.e., the patch $\mathbf{P}_{(u,v)}$ is a 13 × 15 rectangle centered at coordinates $(I^0(u, v), J^0(u, v))$ in the image $\mathbf{X}$.

As an example, for the sensor image Bikes, the four patches in the first row on the detail in Fig. 2 are $\mathbf{P}_{(6,9)}$, $\mathbf{P}_{(6,10)}$, $\mathbf{P}_{(6,11)}$, and $\mathbf{P}_{(6,12)}$ and the coordinates of their centers in the image $\mathbf{X}$ are (64, 127), (64, 141), (63, 155), and (63, 170), respectively.

The 434×541 patches completely cover the 5368×7728 sensor image with slight overlaps between patches due to the slight misalignment of the hexagonal lattice of microlenses with respect to the rectangular lattice of the pixels in the sensor.

The scanning order of the sensor image is defined as follows: the image is traversed patch-by-patch, rowwise, and within each patch the pixels are also traversed rowwise, and we pass to the next patch only after all of the pixels in the current patch are traversed. Due to the selected scanning order, the pixels processed at each patch are those enclosed by the thick black lines in Fig. 2.

A. PREDICTION MODEL

We consider a linear prediction model, where the gray value of the image $\mathbf{X}$ at each pixel from the current patch can be modeled as a linear combination between the gray values of the image at the pixels already scanned in the current patch (a causal intra-template) and between the gray values at *all* pixels of a previously scanned patch (we consider only the patch at the left in the entire experimental section), as shown in Fig. 3. For simplifying the indexing, here we denote by

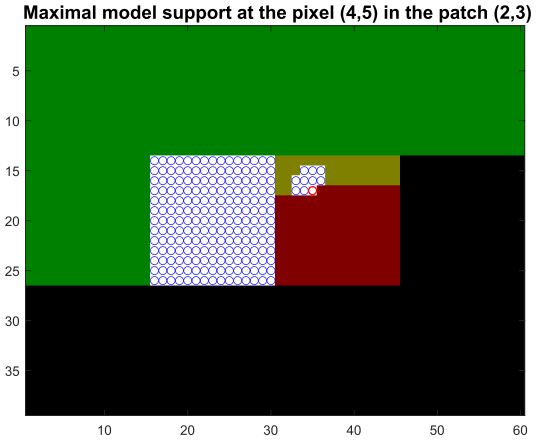

FIGURE 3. The maximum support (encompassing the blue circled pixels) allowed for the sparse filter when applied for prediction at the pixel $(i, j) = (4, 5)$ (shown as a red circle) in the current patch $(u, v) = (2, 3)$. It includes all 195 pixels (denoted as the set $\mathcal{N}$), in the previous (left) patch $(u, v-1) = (2, 2)$, for inter-patch prediction, and includes at most 9 causal pixels in the current patch, forming the set denoted $\mathcal{N}_c(i, j)$, for intra-patch prediction (near the patch border some of the nine pixels are not available).

$\mathbf{P^c} = \mathbf{P}_{(u,v)}$ the current patch to be encoded with the patch index (u, v), and by $\mathbf{P^{cn}} = \mathbf{P}_{(u,v-1)}$ its neighbor patch to the left with the patch index $(u, v-1)$. We refer to the pixels within the patch by the index (i, j) with $1 \le i \le 13$ and $1 \le j \le 15$, so that the gray value $P_{(u,v)}(i, j) = P^c(i, j)$ is identical to the gray value $X(I^0(u, v) + i - 7, J^0(u, v) + j - 8)$. Let us consider that we need to predict $P^c(i, j)$, the gray value at the pixel (i, j) in the current patch, and similarly denote $P^{cn}(i', j')$ the image value at any pixel (i', j') in the neighbor patch.

The prediction model is defined as:

$$\hat{P}_{(u,v)}(i,j) = \sum_{(i',j')\in\mathcal{N}_c(i,j)} P_{(u,v)}(i',j')w_{i'-i,j'-j} + \sum_{(i',j')\in\mathcal{N}} P_{(u,v-1)}(i',j')w^n_{i',j'} + e_{i,j}, \quad (1)$$

where the variables w and w^n with various indices are parameters of the model, and $e_{i,j}$ is the residual of the model. Fig. 3 shows the inter-template $\mathcal{N}$, which is the same for any predicted location (i, j) within the current patch, and the causal intra-template $\mathcal{N}_c(i, j)$, which depends on the location (i, j) of the predicted pixel inside the current patch.

The predicted value is:

$$\hat{P}_{(u,v)}(i,j) = \sum_{(i',j')\in\mathcal{N}_c(i,j)} P_{(u,v)}(i',j')w_{i'-i,j'-j} + \sum_{(i',j')\in\mathcal{N}} P_{(u,v-1)}(i',j')w^n_{i',j'}. \quad (2)$$

For example, considering all of the parameters $\{w, w^n\}$ involved at the pixel (i, j) in the current patch as shown in (1), we will assume that the same linear combination holds true also for the pixel (i, j) at all of the patches that belong to the same class as that of the current patch. This allows us to pool all linear equations for the prediction at the same (i, j) in several patches, resulting in a linear system that can be solved in the least-squares sense. Our solution will be to further constrain the number of nonzero coefficients in (1) and to find a sparse model specific for the location (i, j) in any patch having the same class label ℓ, where useful patch class labeling having M classes is explained in the next section. The degree of sparsity N_s (the maximum number of non-zero coefficients in the set of parameters $\{w, w^n\}$) is fixed in all of our experiments with $N_s = 20$.

Thus, we must design a distinct sparse predictor for each patch class and for each patch pixel location, resulting in $13 \times 15 \times M$ distinct predictors that must be designed and transmitted as additional information to the decoder.

1) SPARSE PREDICTOR DESIGN AND ENCODING THE SPARSE PREDICTOR PARAMETERS

The sparse predictor design is performed similarly to that in [43]: we estimate by least squares (LS) all of the coefficients from (1), sort in descending order the absolute values of the coefficients and keep the first N_{s_1} largest ones, and then declare the corresponding regressors as possibly relevant. Then, we perform an additional least squares estimation, using only the first N_{s_1} selected regressors, and then sort in descending order the absolute values of the N_{s_1} LS coefficients and keep only the largest N_{s_2} of them, again selecting the corresponding regressors as possibly relevant. Then, the same process is repeated once more and finally the desired number of coefficients N_s is obtained, and their corresponding regressors are declared to be the relevant regressors. In the experiments, we use $N_{s_1} = 50$, $N_{s_2} = 35$, and $N_s = 20$. This is a fast process that is much faster than that where the full OLS greedy algorithm is applied as it was done in [11]. We resort to this fast sparse design because we need to perform the sparse design for each pair of (class, patch pixel index) so that the sparse design will be carried out many times.

Here, we describe in more detail the structure of the sparse predictors and the encoding of the predictor parameters.

To present the encoding of the sparse predictor parameters, let us denote by $\mathbf{\Theta}$ the vector of weights (all $\{w, w^n\}$ involved in (1)) of a full predictor to be used at a pixel (i, j), $1 \le i \le 13$, $1 \le j \le 15$ in a patch $\mathbf{P^c}$. The first 195 elements in the vector $\mathbf{\Theta}$ are the inter-patch weights, associated with the gray level values of the pixels in the left neighbor patch, $\mathbf{P^{cn}}$, as shown in (1). The rest of the elements in $\mathbf{\Theta}$ are the weights to be used for the intra-patch part of the predictor (the size of the intra-patch part is at most nine, and depends on the target pixel (i, j) in the patch, as shown in Fig. 3). The sparse prediction design will select only N_s nonzero elements of $\mathbf{\Theta}$, corresponding to the most relevant regressors. These N_s indices form the sparsity mask $\mathcal{I} = \{i_1, i_2, \ldots, i_{N_s}\}$, identifying the non-zero elements of $\mathbf{\Theta}$ for the decoder. The vector formed from the nonzero elements of $\mathbf{\Theta}$ is given by $\boldsymbol{\theta} = \left[\Theta_{i_1}\ \Theta_{i_2}\ \ldots\ \Theta_{i_{N_s}}\right]^T$. Hence, encoding a sparse predictor involves encoding its non-zeros coefficients $\boldsymbol{\theta}$ and the sparsity mask $\mathcal{I}$.

The encoding of the N_s-dimensional vector $\boldsymbol{\theta}$ for each of the $N_P = M \times 13 \times 15$ sparse predictors is performed in N_s groups, where the ℓ'th group has N_P values, consisting of the elements θ_ℓ in all vectors $\boldsymbol{\theta}$. Each non-zero coefficient is uniformly quantized by rounding its fractional part to a certain number of bits (N_{bits}). For a coefficient with absolute value $w = |\theta_\ell|$, the quantized version is $w^q = \lfloor w2^{N_{bits}} \rceil / 2^{N_{bits}}$, which is available to the decoder by transmitting the quantization index for that coefficient that has the integer value $\lfloor w2^{N_{bits}} \rceil$. The sign of θ_ℓ is transmitted by a single bit. The absolute values of the quantization indices are gathered in N_s groups prior to encoding by Golomb-Rice coding. The ℓ'th group is transmitted by Golomb-Rice coding, where the optimal Golomb-Rice parameter is transmitted first, followed by the codes of the elements $\lfloor |\theta_\ell| 2^{Nbits} \rceil$ for all N_P predictors.

The encoding of the sparsity mask $\mathcal{I}$ for all $N_P = M \times 13 \times 15$ sparse predictors is performed by creating an $N_s \times M \times 195$ array with integer elements that is then linearized along the three dimensions (the indices along first dimension change the fastest) and finally the linearized vector is transmitted using the LZ encoding (as implemented in zip encoder).

B. CLASSIFYING PATCHES ACCORDING TO THE BAYER PHASE

Most predictive schemes of lossless compression of CFA images [36]–[38] introduced predictors utilizing the Bayer color labels in the definition of the templates used in the definitions of the contexts and in assigning different predictors to pixels with different Bayer color labels. Similarly, we utilize the regularities and features specific to the Bayer pattern images that clearly make an important contribution in the type of correlation between the pixels in the current patch to encode, $\mathbf{P^c}$, and the corresponding pixels in the neighbor patch, $\mathbf{P^{cn}}$.

Because of the Bayer mask, each pixel in the image sensor acquires the light intensity for only one color from the set $\{R, G, B\}$. For a repetitive 2×2 Bayer pattern, the image has its pixels labeled by the Bayer color attributes from $\{G1, R, B, G2\}$ (the set was renamed as $\{gr, r, b, gb\}$ in the metadata provided by the Lytro camera). Due to the structure of the Bayer mask that has a 2×2 repetitive pattern along the horizontal and vertical directions in the sensor image, each pixel will have an associated Bayer color label, and as in any periodical process, the labels can be identified as the phases of the pixels. In Fig. 4, we show the allocation of the Bayer color labels to the sensor image pixels, and the related definition of the integer labels called in the sequel simply phases. An almost identical convention was used in [44] for defining the decomposition of the sensor image into four polyphase components where the pixels belonging to the same polyphase component have the same color label. The polyphase index in Fig. 4 is identical to the convention from [44] when one reads as integers 0,1,2,3 the binary strings 00,01,10,11 that are the pixel labels in [44].

For the pixel at location (I, J) in the sensor image, we call the attribute denoted $\varphi(I, J) \in \{0, 1, 2, 3\}$ the *Bayer phase*

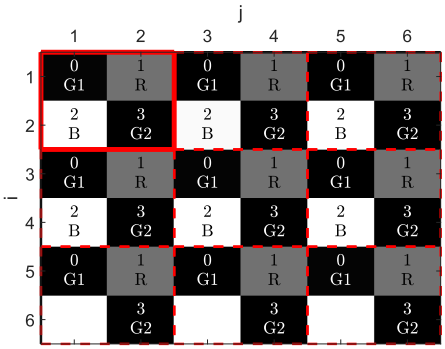

FIGURE 4. The Bayer phase $\varphi \in \{0, 1, 2, 3\}$ and color label associated with each pixel of the CFA image by repeating the 2×2 basic cell (in bold red borders), starting from the upper-left corner of the image.

that indicates to which of the four polyphase components the pixel belongs; the *Bayer phase* is given by:

$$\varphi(I, J) = 2((I - 1) \mod 2) + ((J - 1) \mod 2), \quad (3)$$

leading to the phases illustrated in Fig. 4. We note that the row and column indices I and J start from 1, as in MATLAB, and hence the top left corner pixel (1, 1) of the image receives the label phase 0.

Inside the SRRC algorithm, we only need to group the pixels in the four classes by their phase label (a number in {0,1,2,3}), and it is not necessary to specify the correspondence of the phase label to the colors. Fig. 4 was constructed based on the information that the label of the upper left pixel is a "G1".

Taking the example of the first row of four patches $(u, v) = (6, 9), (6, 10), (6, 11), (6, 12)$ depicted in Fig. 2 for which the centers are located at $(I^0(u, v), J^0(u, v)) = (64, 127), (64, 141), (63, 155), (63, 170)$, we obtained the phases $\varphi = 2, 2, 0, 1$, respectively. This means that the central pixel in patch (6, 9) is blue, that of patch (6, 10) is blue, that of patch (6, 11) is green, and that of patch (6, 12) is red.

The dependences needed in (1) will be obtained from the analysis of the equivalent situations: whenever the current patch $\mathbf{P^c}$ has a phase φ^c at its center, and its left patch $\mathbf{P^{cn}}$ has a phase φ^{cn} at its center, we expect the same type of dependence to hold for a given (i, j) in (1). Therefore, we consider all 16 possible pairs $(\varphi^c, \varphi^{cn})$ as possibles classes, and say that a patch $\mathbf{P^c}$ belongs to the specific class $(\varphi^c, \varphi^{cn})$.

More formally, let us consider the current patch, $\mathbf{P^c}$, indexed by (u, v), and its neighbor patch, $\mathbf{P^{cn}}$, indexed by $(u, v-1)$. The phase at the center of $\mathbf{P^c}$ is $\varphi(I^0(u, v), J^0(u, v))$ and the phase at the center of $\mathbf{P^{cn}}$ is $\varphi(I^0(u, v-1), J^0(u, v-1))$ as in (3). We define the phase signature of the current patch, $\mathbf{P^c} = \mathbf{P}_{(u,v)}$, as the variable $L_\varphi(u, v) = 1 + \varphi(I^0(u, v), J^0(u, v)) + 4\varphi(I^0(u, v-1), J^0(u, v-1))$, which is an integer between 1 and 16.

A fruitful modeling assumption is that for a given phase signature $L_\varphi(u, v) = L_\varphi$, the possible predictive dependences between the pixels of $\mathbf{P}_{(u,v-1)}$ and those of $\mathbf{P}_{(u,v)}$ are similar,

while for a different phase signature, the dependences will have to be inferred separately because they will refer to the relation between the pixels at different color labels.

We name the algorithm that designs the sparse predictors by assigning the class label $L_\varphi(u, v)$ to the patch $\mathbf{P}_{(u,v)}$ SRRC-PHASE. As shown below, this algorithm displays excellent performance.

C. CLASSIFYING PATCHES ACCORDING TO THE DEPTH INFORMATION

To avoid the performance of the predictor design too many times, it is useful to pool the regression equations (1) over as few classes of patches as possible. While the pooling in the previous section was based on the phase signature at a patch, $L_\varphi(u, v)$, based on the Bayer mask colors, now we consider the second natural choice, namely, of pooling the patches together depending on the value of the depth in the scene (or according to the disparity) at the central pixel of a patch.

The depth estimation from rectified LF array data has been studied intensely and has a dedicated evaluation benchmark with its own datasets and evaluation methodology [45] with more than 70 methods evaluated under various metrics. According to most of these metrics, methods based on convolutional neural networks are among the top methods.

In the experimental section, we first present experiments with the depthmap images saved in the EPFL Lytro database [25] available at the JPEG Pleno Database website, namely, the EPFL Light-field data set [46] generated according to [25] by the Lytro proprietary software for depth estimation. Then, we also show a comparison with the results obtained using the depthmap estimates obtained by the recent EPINET convolution neural network depth estimator [47], that was trained in [48] on the light field datasets from HCI Benchmark [45] and then was applied to the Lytro datsets of the JPEG Pleno Light Field CTC [46]. Visually, the depthmap produced by EPINET is less noisy and conveys finer details than the depthmap produced by the Lytro proprietary software as observed for the data that we made available in [49].

The depthmap is needed in SRRC at a resolution of $(N_r \times N_c)$, and if the depth estimation routine produces a different resolution, it is necessary first to resize the depthmap to a size of $(N_r \times N_c)$.

The depthmaps are rather noisy and they do not convey very precise values, so that we decided to quantize the values using a relatively small number of intervals, M, also in accordance with our goal of not having too many classes for patches. A study of the performance obtained when changing M is shown in the experimental section. In fact, we need only to classify the patches (into a small number of classes), and hence the scale of the depth value or disparity value is not important. We first linearly rescale the range of the depth values provided by any of the two depth estimators, so that the depth value 0 is the closest point in the scene and depth value 1 is the furthest away, obtaining a $(N_r \times N_c)$ depth image that we denote $\mathbf{D}$. Then, we transform the values of $\mathbf{D}$ by scaling and rounding with the relation $D_q(u, v) = 1 + \lfloor D(u, v)(M - 1) \rceil$, obtaining for each pixel an integer class label between 1 and M.

After this process, the number of elements in some classes may turn out to be too small, and therefore we performed an additional reassignment of the less populated classes as follows. We traverse the classes from 1 to $(M - 1)$, and if class ℓ has less than N_{min} elements, we merge its elements to the class $\ell + 1$. Then, we finally traverse the classes from M down to 2 and for any class ℓ with less than N_{min} elements, we merge its elements to the class $\ell - 1$. The value of N_{min} was taken as 400. After the reassignment, the final number of distinct classes may be less than the initial value M, and then we let M denote the true number of classes, after reassignment.

The $(N_r \times N_c)$ image of labels must be transmitted to the decoder and for this task, we use the CERV encoding method [50] that is well-suited for transmitting depthmap images with sharp transitions between piecewise constant regions.

In the SRRC-DEPTH method, we use $L_D(u, v) = D_q(u, v)$ as a label of the patch $\mathbf{P}_{(u,v)}$. The results of using this labeling of patches for the sparse predictor design are shown in the experimental section; however, it is observed that the labeling based on the phase signature at a patch, $L_\varphi(u, v)$ almost always produces better results.

D. CONTEXTING POLICY

The context at a pixel in most lossless predictive coding methods for gray level images is selected to depend on the quantized value of the estimated energy of the prediction error at that pixel. The difference between the context definitions lies in the manner in which the energy of the error is estimated. An often used approximation is that the context is the uniform quantized logarithm of the mean absolute fluctuations of the signal in a causal neighborhood of the current prediction point. The M-MRP method [16] has shown excellent performance by obtaining the context through weighing the absolute values of residuals at the location of regressors, where the weighing values are the Euclidean distances between the geometrical locations of the regressor pixel and the current pixel.

In our codec, we use the magnitudes of the sparse predictor vector elements (all $\{w, w^n\}$ used in (1)) as the relevant weights for the residuals, when defining the context, since these magnitudes convey the information about the relevance of each regressor retained in the final sparse predictor mask (out of the 204 possible regressors). Intuitively, the magnitude of an LS coefficient resulting from the sparse design can be considered as a measure of the *relevance* of the corresponding regressor pixel, and we define the contexts based on this type of relevance (which is also reflected in the name of the method, sparse relevant regressors and contexts (SRRC)).

When encoding an integer prediction residual $R_{(u,v)}(i, j) = P_{(u,v)}(i, j) - \lfloor \hat{P}_{(u,v)}(i, j) \rceil$ at location (i, j) within the patch $\mathbf{P}^c$, the context is selected as a function of the normalized error

amplitude defined as:

$$E_{i,j} = \left(\sum_{(i',j')\in\mathcal{N}_c(i,j)} |R_{(u,v)}(i',j')| \cdot |w_{i'-i,j'-j}| + \sum_{(i',j')\in\mathcal{N}} |R_{(u,v-1)}(i',j')| \cdot |w^n_{i',j'}| \right) \frac{1}{\|\mathcal{W}_{i,j}\|_1} \quad (4)$$

where $\|\mathcal{W}_{i,j}\|_1$ is the L^1-norm of the (sparse) vector with elements $\{w\}$ (the nine causal weighting coefficients for intra-patch elements) and $\{w^n\}$ (195 weighting coefficients for the inter-patch elements). By the sparse design, only N_s elements have non-zero values. Since the sparse filter selected only N_s relevant regressors out of all of the possible regressors with each regressor influencing the prediction in (1) with their corresponding weighing factor, we decide to use the same weighing factors as in the prediction model (1) also in the average amplitude definition in order to form the weighted L^1-norm of the residuals at the locations used for prediction. Thus, $E_{i,j}$ is a proxy for the estimation of the normalized mean absolute error expected for the current prediction. Following the traditional definition of the contexts, we use the rounded value of the base2 logarithm of $E_{i,j}$ as an integer context, $C_{(u,v)}(i,j) = \lfloor \log_2 E(i,j) \rceil$, which is the definition of the context used in all experiments.

The final stage of the encoder is the arithmetic coding. As defined earlier, we traverse the lenslet image patch-by-patch and for each patch $\mathbf{P}_{(u,v)}$ and pixel (i,j) inside the patch, we encode the integer prediction residual $R_{(u,v)}(i,j)$ using the cumulative frequency table collected at the context $C_{(u,v)}(i,j)$, using the arithmetic coding package from [51]. The decoder can track the values of the context identically to the encoder, resulting in the lossless reconstruction of $R_{(u,v)}(i,j)$ that together with the integer prediction, $\lfloor \hat{P}_{(u,v)}(i,j) \rceil$, produces the lossless reconstruction $P_{(u,v)}(i,j) = \lfloor \hat{P}_{(u,v)}(i,j) \rceil + R_{(u,v)}(i,j)$.

III. EXPERIMENTAL RESULTS

We used images from the database introduced in [25] and available on-line at [46]. We used the 12 plenoptic scenes denoted $I_{01}, \ldots, I_{12}$, as shown in Table 1. The camera sensor images are gray-level images of size 5368×7728 with a resolution of 10 bits per pixel. For each scene, we consider two important sensor images provided at [46]: first, the input sensor image of the scene that we denote $\mathbf{X}$ that is contained in a LFR file, and second, the white image $\mathbf{W}$ that is the plenoptic camera sensor image obtained when taking the image of a white sheet using the same camera setting as when imaging the real scene image $\mathbf{X}$. The two images are needed in the plenoptic processing chain for obtaining the light field array of views. The format and procedures for reading the images $\mathbf{X}$ and $\mathbf{W}$ associated with each scene are given in the Light field MATLAB toolbox [41] and are also provided for convenience in the software implementation of our methods provided at [49].

TABLE 1. Set of twelve scenes used in the experiments.

Identifier	Image name
I_{01}	Bikes
I_{02}	Danger_de_Mort
I_{03}	Flowers
I_{04}	Stone_Pillars_Outside
I_{05}	Vespa
I_{06}	Ankylosaurus_&_Diplodocus_1
I_{07}	Desktop
I_{08}	Magnets_1
I_{09}	Fountain_&_Vincent_2
I_{10}	Friends_1
I_{11}	Color_Chart_1
I_{12}	ISO_Chart_12

A. SELECTION OF THE PARAMETERS OF THE SPARSE PREDICTORS FOR THE SRRC-PHASE ALGORITHM

The main parameters of a sparse predictor are the degree of sparsity, N_s, defined as the number of nonzero coefficients selected out of the number of all possible regressors included in the prediction template, and the number of bits per fractional part of a coefficient, N_{bits}.

The degree of sparsity clearly affects the performance of the SRRC scheme. Fig. 5 shows the lossless bitrate for SRRC-PHASE in bpp for the sensor image I_{01}-Bikes for several degrees of sparsity between 5 to 50 and for the number of bits per fractional part of predictor coefficient N_{bits} in the set $\{8; 12; 16\}$. It is observed that a too low N_s has a strong impact, e.g., at $N_s = 5$, the lossless bitrate is approximately 7% worse than the best value that is achieved at $N_s = 30$. There is no reason to choose a too low N_s because this will lead to less accurate prediction and to large residuals, resulting in a poor compression performance.

On the other end, a too high N_s, e.g., $N_s = 50$ will require more bits to encode the predictors without a significant reduction in the bitrate needed for encoding the residuals.

It is observed from Fig. 5 that for Bikes the range between $N_s = 15$ to $N_s = 50$ is rather flat, incurring changes of the lossless bitrate of less than 0.7%.

The number of fractional bits per coefficients, N_{bits}, has a weaker impact on the compression performance. For a given N_s, the lossless bitrate in bpp is observed to be within 0.3% from the best value achieved at that N_s. Hence, a very careful selection of N_{bits} is expected to have little influence on improving the performance of the SRRC-PHASE scheme.

Since performing such a study for selecting optimal N_s and optimal N_{bits} for each dataset requires running the encoding algorithm multiple times, we decided to use the fixed values $N_s = 20$ and $N_{bits} = 12$ for all of the datasets in our experiments. As a final check, we show the behavior of the compression rate for I_{04}-Pillars, for varying N_s, at $N_{bits} = 12$, as the bottom curve of Fig. 5. It is observed that at our selected fixed value, $N_s = 20$, the bitrate is 5.125, which is approximately 0.45% worse than the best bitrate for this dataset, which is 5.102 at $N_s = 30$. Hence, we may expect small slight gains when changing the parameter $N_s = 20$, but we would like to maintain a fair scenario in the experiments

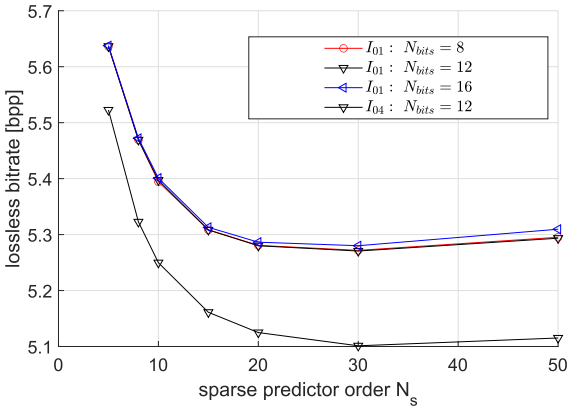

FIGURE 5. Coding results for SRRC on camera sensor images I_{01}-Bikes and I_{04}-Pillars for various degrees of sparsity N_s and number of bits N_{bits} per fractional part of a predictor coefficient.

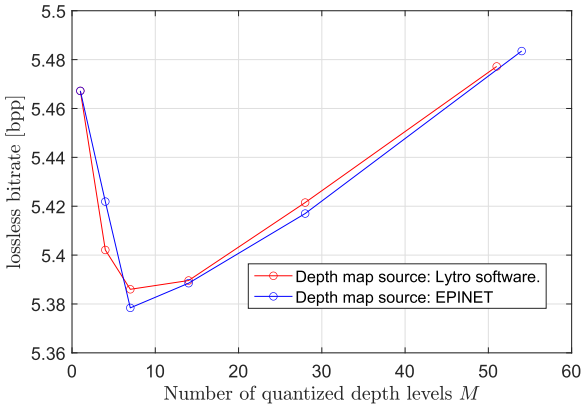

FIGURE 6. Comparing the bitrate for the lossless encoding of 'Bikes' image sensor X by SRRC-DEPTH method using the quantized depth maps provided by Lytro proprietary software used in [25] and by Epinet [47], for different number of quantization levels M.

by using the same N_s and N_{bits} for all of the datasets and for both SRRC-PHASE and SRRC-DEPTH.

B. INSIGHTS INTO SRRC-DEPTH METHOD USING TWO DIFFERENT ESTIMATES OF DEPTHMAPS AND SEVERAL DEPTH QUANTIZATION LEVELS

We have run the algorithm SRRC-DEPTH for the camera sensor image I_{01}-Bikes, first using the depthmap, provided in the original EPFL database [25], estimated by a Lytro proprietary software. Then, we have run the algorithm using the depthmap obtained from the EPINET estimate. For both depthmaps we have applied the pre-processing and quantization method from Section II-C, for several numbers of quantization levels, M. The resulting performance is shown in Fig. 6. Surprisingly, the lossless bitrates obtained with the two depthmap estimates are very close to each other, for a same value of M. An examination of the dependence of the bitrates on M shows that the values at $M = 8$ and $M = 16$ are rather close. We have selected to run the SRRC-DEPTH in the rest of the paper on all datasets $I_{01}, \ldots, I_{12}$ using the value of $M = 16$ with the same number of classes as in SRRC-PHASE for a fair comparison. Additionally, we chose to use the original Lytro dataset estimate of the depthmap in the rest of the experiments, since it is readily available for all of the datasets in the database [25], while the EPINET depthmap estimation requires the execution of the CNN estimator on GPU.

Table 5 of Supplemental material shows the breakdown of the final bitrate into individual bitrates needed by the various elements that are encoded in the final archive produced by SRRC-DEPTH. Once again, it is observed that SRRC-DEPTH shows very similar behavior for the two different depthmap estimates. The fraction of the bitrate for encoding the depthmap itself by CERV is very small, hinting that the differences observed in the next subsection between the bitrates of SRRC-PHASE and SRRC-DEPTH are not due to the extra cost of encoding the depthmap, but rather arise from the better classification of the patches obtained by L_φ.

Table 6 of Supplemental material compares the performance when encoding five datasets by SRRC-DEPTH method using for each dataset the corresponding depthmap from the set of depthmaps $\mathcal{D}_1$ obtained by Epinet [47] and from the set of depthmaps $\mathcal{D}_2$ from [25], when the targetted number of quantization levels is $M = 16$. The compression performance at each datasets is seen to be very similar for the two depthmap estimation methods.

C. COMPARISONS BETWEEN THE PERFORMANCE OF SRRC-PHASE, SRRC-DEPTH, AND OF OTHER CODECS

Columns 2 and 3 of Table 2 show the bitrates achieved by the proposed algorithm, SRRC, in the two versions, namely, when the Bayer phase L_φ at the neighbor patches is used for classifying the patches and when the quantized depth L_D is used for labeling the patches. The first version, SRRC-PHASE, is consistently better than the second, SRRC-DEPTH, on average by 0.19 bpp. The values presented in Table 2 are the compressed file sizes in bits divided by the total number of pixels in the input sensor image. The file sizes are the actual sizes from the file system, and the lossless reconstruction after decoding is confirmed for all of the files.

Table 2 also shows the results of the specialized coder that we call sparse modeling compression (SMC), which is the method presented in [8], and we observe an improvement by 0.58 bpp on average for our SRRC based on L_φ. We also show the results of a general use coder for color and gray images based on the FLIF (free lossless image format based on maniac compression), method introduced in [52]; it is observed that for all files, SRRC obtains better compression rate than FLIF, improving the coding performance by 0.53 bpp on average. Compared to the standard lossless JPEG 2000 coding, SRRC shows improvements of approximately 1.17 bpp on average.

We also present a comparison of SRRC with respect to one of the best-performing CFA image compression methods, namely, the HPCM [37] method, the results for which are shown in column 6 (we have used our own implementation of

TABLE 2. Coding results for camera sensor images **X** and **W** by SRRC-PHASE, SRRC-DEPTH and other codecs.

Image	Scene Input Image $\mathbf{X}$							White Image $\mathbf{W}$				
Method[a]	SRRC_X Phase	SRRC_X Depth	FLIF_X	SMC_X	HPCM_X	JP2_X	JLS_X	SRRC_W Phase	FLIF_W	HPCM_W	JP2_W	JLS_W
I_{01}	**5.28**	5.39	5.79	5.76	5.79	6.24	6.56	**4.91**	6.17	7.04	7.83	8.47
I_{02}	**5.04**	5.14	5.41	5.57	5.34	5.74	6.32	**5.05**	6.25	6.93	7.64	8.36
I_{03}	**5.23**	5.26	5.61	5.79	5.52	5.93	6.32	**5.05**	6.25	6.93	7.64	8.36
I_{04}	**5.12**	5.27	5.66	5.84	5.66	6.02	6.54	**4.96**	6.23	6.92	7.66	8.38
I_{05}	**4.83**	4.89	5.16	5.17	5.33	5.7	5.88	**4.81**	6.02	7.21	7.99	8.42
I_{06}	**4.98**	5.30	5.72	5.71	6.1	6.69	6.85	**4.82**	6.06	7.14	7.95	8.47
I_{07}	**5.94**	6.14	6.53	6.55	6.74	7.24	7.86	**4.93**	6.21	6.97	7.73	8.42
I_{08}	**5.07**	5.35	5.76	5.72	6.15	6.79	6.87	**4.82**	6.06	7.14	7.95	8.47
I_{09}	**5.4**	5.49	5.81	5.94	5.99	6.49	6.65	**4.88**	6.11	7.12	7.93	8.47
I_{10}	**5.13**	5.37	5.75	5.76	5.82	6.23	6.97	**4.93**	6.21	6.97	7.73	8.42
I_{11}	**4.6**	5.00	5.21	5.13	5.59	6.08	6.4	**4.67**	5.81	7.24	8	8.22
I_{12}	**5.21**	5.50	5.77	5.79	6.13	6.67	6.5	**4.88**	6.11	7.12	7.93	8.47
Average	**5.15**	5.34	5.68	5.73	5.85	6.32	6.64	**4.89**	6.12	7.06	7.83	8.41

All results are compressed file sizes per pixel with the number of pixels equal to 5368×7728.
[a]SRRC (sparse relevant regressors and contexts) - proposed method; FLIF method from [52]; SMC- method from [8]; HPCM- method from [37]; JP2 - JPEG 2000 [53], [54]; JLS - JPEG - LS [55].

the HPCM method); it is observed that SRRC-PHASE shows better performance than HPCM by 0.7 bpp on average.

It is found that our best proposed method, SRRC-PHASE, uses in average 22.4% less bits than JPEG-LS, 18.5% less bits than JPEG 2000, and 12% less bits than HPCM [37] on the sensor images **X** for the 12 datasets considered in the experimental section.

This is consistent with the above-described results for the compression efficiency of the CFA specialized methods that are superior to the general lossless compression methods, e.g., the HPCM method [37] uses on average 24% less bits than JPEG-LS and 10% less than JPEG 2000 for the Kodak datset that is a common dataset used for comparing CFA compression methods. The competition between the CFA specific methods is closer, e.g., HPCM successfully encodes the Kodak dataset with 2% less bits than CMBP [38] and 7% less bits than LCMI [35]. The fact that SRRC-PHASE uses 12% less bits than HPCM for the plenoptic camera sensor images shows that in addition to effectively taking advantage of the Bayer pattern regularities, SRRC-PHASE also highly effectively utilizes the regularities due to the array of microlenses.

The rightmost five columns of Table 2 show the results of compressing **W**, the "white" sensor image associated with the used settings for the plenoptic camera when capturing each scene by using all codecs (the results for the SMC methods were unavailable). We did not run SRRC-DEPTH for **W** because this lenslet image corresponds to a scene with constant depth, so that no gains can be obtained by using the depthmap (which was not even available). The database of white images associated with the camera used for collecting the lytro dataset is also found on the calibration data given for each of the two grand challenges, and is available on-line at [46].

Comparison to the competing methods shows that SRRC-PHASE obtains much larger gains on **W** with respect to the JPEG-LS and JPEG-2000 general codecs, when compared to the gains obtained by SRRC-PHASE on **X**. Additionally, the gains with respect to HPCM are larger on **W**.

It is observed by the compressed size that some scenes such as I_{02} and I_{03} have the same associated white images. The information content in the bits for each of these white images is remarkably large and has almost the same information size as the scene themselves. The white image information plays an essential role in obtaining the final LF array of views because in the plenoptic processing pipeline the two sensor images, **X** and **W** are used for generating the devignetted image, by pixelwise divisions $X(i,j)/W(i,j)$ at all 5368×7728 pixels.

D. COMPLEXITY OF SRRC-PHASE ALGORITHM

Here, we present the running times of the programs used to produce the results shown in the experimental section. The programs are partly written in MATLAB and partly in C (the arithmetic coding part) and are available online at [49]. We present the running times on a laptop that has a i7-7820HQ CPU @ 2.90 GHz processor, 64 GB RAM, and 64-bit operating system. The times are mostly indicative of the expected running times because they vary depending on the other tasks performed by the computer (we noticed differences of as much as 10% when running the same program twice on the same computer). Because we make the source program publicly available, the execution times can be obtained on other platforms when needed. For the memory usage of the programs, we use amounts in bytes, MB, and GB, using the definitions $1\text{MB} = 10^6$ bytes and $1\text{GB} = 10^9$ bytes.

Columns 2 and 6 of Table 3 show the total encoding time and the total decoding time, respectively, for running the SRRC-PHASE algorithm on the twelve datasets. We also show the breakdown into component times: T_{ENC_1} is the time required for running the sparse prediction design and

TABLE 3. Encoding and decoding times for SRRC on camera sensor images X.

Times[a]	T_{ENC}	T_{ENC_1}	T_{ENC_2}	T_{ENC_3}	T_{DEC}	T_{DEC_1}	T_{DEC_2}
File	Encoding Times [s]				Decoding Times [s]		
I_{01}	1345	817	376	146	695	263	408
I_{02}	1275	811	379	80	575	144	406
I_{03}	1313	810	380	117	646	214	408
I_{04}	1276	825	379	66	547	117	407
I_{05}	1407	827	400	175	754	322	409
I_{06}	1192	773	360	52	495	91	382
I_{07}	1235	768	362	99	580	180	382
I_{08}	1218	777	363	73	530	129	379
I_{09}	1281	765	362	148	712	278	412
I_{10}	1385	909	394	76	568	139	409
I_{11}	1215	778	357	76	544	137	386
I_{12}	1200	776	359	61	507	108	379
Average	1279	803	373	97	596	177	397

[a] T_{ENC} (overall encoding time); T_{ENC_1} (design of sparse predictors); T_{ENC_2} (computing the residuals);T_{ENC_3} (encoding of residuals); T_{DEC} (overall decoding time); T_{DEC_1} (decoding of residuals), T_{DEC_2} (computing the predictions).

encoding the predictor parameters; T_{ENC_2} is the time required for computing the predictions and residuals; and T_{ENC_3} is the time required for encoding of residuals. At the decoder side, the major components are T_{DEC_1} for decoding the residuals and T_{DEC_2} for computing the predictions. On average, the encoding takes 21 minutes and decoding takes 10 minutes.

It is not easy to experimentally trace the needed memory usage of the programs because the programs are running in MATLAB, and the MATLAB core itself needs some substantial amount of memory (2-3 GB). We start by presenting the memory requirements for the major data structures that are used in the encoder of the SRRC-PHASE method (the best performing of the two SRRC variants), following the blocks of the algorithm in Fig. 1. The input sensor image has $M_r \times M_c$ elements, stored with 2 bytes per pixel, in a 83MB data structure; however, along the processing, they are in some cases converted for high-precision computations to double format (8 bytes), hence requiring 332MB. The $N_r \times N_c$ matrices $\mathbf{I^0}$, $\mathbf{J^0}$, for storing the coordinates of centers of the microlenses in the sensor image grid, require 3.8 MB.

The dominant memory consumption is certainly that required during the sparse predictor design process for collecting the data matrices needed in the LS designs involved in the sparse design procedure, as presented in Section II-A1.

In each sparse design, say at patch class L_φ and at a patch pixel index (i, j), we need to run a procedure starting with solving in the LS sense a system $\mathbf{b} = \mathbf{A\Theta}$, where the $(n_A \times m_A)$ data matrix $\mathbf{A}$ contains in each row the regressors from (1) (see Section II-A1), hence each element of $\mathbf{A}$ can be stored in 2 bytes. The number of rows n_A is the number of patches with the patch class L_φ. The classes L_φ with the largest number of patches are the classes $L_\varphi = 0, 5, 10, 15$ (for the pairs $(\varphi^c, \varphi^{cn}) = (0, 0), (1, 1), (2, 2), (3, 3)$) with about $N_{max.Patch} = 41000$ patches each. The largest number of regressors is $m_A = 195 + 9$ (as shown in Fig. 3).

The sparse designs are performed iteratively in 16 iterations, where in a single iteration, we collect the data matrices $\mathbf{A}$ and $\mathbf{b}$ for all of the predictors needed at the patches of class L_φ, namely, the predictors for each of the 195 distinct pixels (i, j) of a patch (we chose this arrangement to lower the memory requirements).

Hence, the worst case of required memory during the sparse design procedure (largest sizes n_A, m_A) is for $L_\varphi = 0, 5, 10, 15$, where for each of the 195 patch pixel indices we need to store a data matrix $\mathbf{A}$, with at most $N_{max.Patch} \cdot m_A \cdot 2$ bytes. For all 195 patch pixel indices, the total memory requirement is $N_{max.Patch} \cdot m_A \cdot 2 \cdot 195 = 41000 * 204 * 195 * 2 = 3.27 * 10^9$ bytes.

Using the *memory* and *whos* functions of MATLAB, we experimentally evaluated the memory usage at the various stages of the algorithm reported in Table 4. The second column reports the memory occupied by the variables created by the user (collected by adding up the sizes of all of the variables reported by *whos*), while the third column presents the overall memory required from the operating system by MATLAB (including the MATLAB's system own created variables), as reported by the *memory* function. It is observed that the encoder required memory is dominated by the memory for sparse predictor design that has the total value of $4.1 \cdot 10^9$ bytes (out of which the data matrices occupy $3.3 \cdot 10^9$ bytes, as described above). By modifying the details of the implementation, the memory requirements can be reduced for example at the expense of larger encoding times. If the programs will be re-coded into C, using similar variables and data structures, the memory requirements most likely will remain similar to those reported in column 2 of Table 4; however, the extra space up to the values in column 3, accounting for MATLAB kernel variables, will no longer be needed. Nevertheless, the overall requirements will remain approximately 4 GB at the encoder and approximately 3 GB at the decoder.

Finally, in the Supplemental material, we report the execution times for the SRRC and the general codecs used in the compression performance comparisons.

TABLE 4. Memory used in the SRRC-PHASE algorithm in MB (i.e., in 10^6 bytes).

	Memory for variables	Memory overall
Encoder		
Input data and initialization	758	3233
Sparse predictor design and prediction coefficients encoding	4103	7137
Prediction and residual computation and encoding	2536	5618
Decoder		
Initialization	676	3385
Decode residuals and predictors coefficients	1173	3908
Prediction and reconstruction of original $\mathbf{X}$	2949	5679

When reporting the times for SRRC-DEPTH, we do not report the times for acquiring the depthmap, since they vary strongly depending on the program used for depth estimation. For example, the EPINET program trained on the HCI data can provide a depthmap estimate for a Lytro dataset in a couple of seconds but requires a GPU for execution and a large amount of memory for storing the CNN parameters. The extra call of CERV [50] encoding and decoding requires only a couple of seconds for the 434×541 quantized depth images).

IV. CONCLUSION

We have presented several compression methods for encoding plenoptic camera sensor images. The best results are obtained by SRRC-PHASE that utilizes both the regularities induced by the Bayer mask located in front of the image sensor, and the regularities due to the array of microlenses. Compared to the methods developed for traditional camera CFA images, the proposed methods significantly reduce the size of the compressed file due to utilization of the regularities induced by the array of microlenses that are efficiently exploited by the sparse predictor design and by the tuned contexts used in the arithmetic coding of the residuals.

REFERENCES

[1] C. Conti, L. D. Soares, and P. Nunes, "Dense light field coding: A survey," *IEEE Access*, vol. 8, pp. 49244–49284, 2020.
[2] T. Ebrahimi, S. Foessel, F. Pereira, and P. Schelkens, "JPEG Pleno: Toward an efficient representation of visual reality," *IEEE Multimedia Mag.*, vol. 23, no. 4, pp. 14–20, Oct. 2016.
[3] P. Schelkens, P. Astola, E. A. B. D. Silva, C. Pagliari, C. Perra, I. Tabus, and O. Watanabe, "JPEG Pleno light field coding technologies," *Proc. SPIE*, vol. 11137, Sep. 2019, Art. no. 111371G.
[4] M. Wien, J. M. Boyce, T. Stockhammer, and W.-H. Peng, "Standardization status of immersive video coding," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 5–17, Mar. 2019.
[5] C. Perra, "Lossless plenoptic image compression using adaptive block differential prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1231–1234.
[6] P. Helin, P. Astola, B. Rao, and I. Tabus, "Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression," in *Proc. 3DTV-Conf., True Vis.-Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jul. 2016, pp. 1–4.
[7] R. Zhong, S. Wang, B. Cornelis, Y. Zheng, J. Yuan, and A. Munteanu, "L1-optimized linear prediction for light field image compression," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2016, pp. 1–5.
[8] I. Tabus and P. Helin, "Microlens image sparse modelling for lossless compression of plenoptic camera sensor images," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2017, pp. 1907–1911.
[9] P. Helin, P. Astola, B. Rao, and I. Tabus, "Minimum description length sparse modeling and region merging for lossless plenoptic image compression," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 7, pp. 1146–1161, Oct. 2017.
[10] I. Schiopu, M. Gabbouj, A. Iosifidis, B. Zeng, and S. Liu, "Subaperture image segmentation for lossless compression," in *Proc. 7th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Nov. 2017, pp. 1–6.
[11] I. Tabus, P. Helin, and P. Astola, "Lossy compression of lenslet images from plenoptic cameras combining sparse predictive coding and JPEG 2000," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4567–4571.
[12] I. Schiopu, M. Gabbouj, A. Gotchev, and M. M. Hannuksela, "Lossless compression of subaperture images using context modeling," in *Proc. 3DTV Conf., True Vis.-Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jun. 2017, pp. 1–4.
[13] J. M. Santos, P. A. A. Assuncao, L. A. D. S. Cruz, L. Tavora, R. Fonseca-Pinto, and S. M. M. Faria, "Lossless light-field compression using reversible colour transformations," in *Proc. 7th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Nov. 2017, pp. 1–6.
[14] I. Schiopu and A. Munteanu, "Macro-pixel prediction based on convolutional neural networks for lossless compression of light field images," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 445–449.
[15] A. Miyazawa, Y. Kameda, T. Ishikawa, I. Matsuda, and S. Itoh, "Lossless coding of light field camera data captured with a micro-lens array and a color filter," in *Proc. Int. Workshop Adv. Image Technol. (IWAIT)*, Jan. 2018, pp. 1–4.
[16] J. M. Santos, P. A. A. Assuncao, L. A. D. S. Cruz, L. M. N. Tavora, R. Fonseca-Pinto, and S. M. M. Faria, "Lossless compression of light fields using multi-reference minimum rate predictors," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 408–417.
[17] J. M. Santos, P. A. A. Assuncao, L. A. D. S. Cruz, L. M. N. Tavora, R. Fonseca-Pinto, and S. M. M. Faria, "Lossless compression of light fields using multi-reference minimum rate predictors," in *Proc. Joint Pictures Expert Group, 82nd JPEG Meeting*, Lisbon, Portugal, Jan. 2019, pp. 408–417.
[18] I. Schiopu and A. Munteanu, "Deep-learning-based macro-pixel synthesis and lossless coding of light field images," *APSIPA Trans. Signal Inf. Process.*, vol. 8, pp. 1–12, Jul. 2019.
[19] M. U. Mukati and S. Forchhammer, "EPIC: Context adaptive lossless light field compression using epipolar plane images," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2020, pp. 43–52.
[20] M. Rizkallah, T. Maugey, and C. Guillemot, "Prediction and sampling with local graph transforms for quasi-lossless light field compression," *IEEE Trans. Image Process.*, vol. 29, pp. 3282–3295, 2020.
[21] M. Rizkallah, T. Maugey, and C. Guillemot, "Graph-based spatio-angular prediction for quasi-lossless compression of light fields," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 379–388.
[22] R. Zhong, I. Schiopu, B. Cornelis, S.-P. Lu, J. Yuan, and A. Munteanu, "Dictionary learning-based, directional, and optimized prediction for lenslet image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 4, pp. 1116–1129, Apr. 2019.
[23] M. Rerabek, T. Bruylants, T. Ebrahimi, F. Pereira, and P. Schelkens, "Call for proposals and evaluation procedure, ICME 2016 grand challenge: Light-field image compression," École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, Tech. Rep., Jul. 2016. [Online]. Available: https://www.epfl.ch/labs/mmspg/wp-content/uploads/2019/01/CFP.pdf
[24] *JPEG Pleno Call for Proposals on Light Field Coding*, Standard ISO/IEC JTC 1/SC29/WG1 JPEG, Doc. N74014, Jan 2017.
[25] M. Rerabek and T. Ebrahimi, "New light field image dataset," in *Proc. 8th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, 2016, pp. 1–2.
[26] I. Viola and T. Ebrahimi, "Quality assessment of compression solutions for ICIP 2017 grand challenge on light field image coding," in *Proc. 9th Workshop Hot Topics 3D Multimedia (Hot3D)*, Jul. 2018, pp. 1–6.
[27] S. Zhao and Z. Chen, "Light field image coding via linear approximation prior," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4562–4566.
[28] W. Ahmad, R. Olsson, and M. Sjostrom, "Interpreting plenoptic images as multi-view sequences for improved compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4557–4561.
[29] M. B. D. 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.

[30] H. S. Malvar and G. J. Sullivan, "Progressive-to-Lossless compression of color-filter-array images using macropixel spectral-spatial transformation," in *Proc. Data Compress. Conf.*, Apr. 2012, pp. 3–12.
[31] Y. Lee, K. Hirakawa, and T. Q. Nguyen, "Camera-aware multi-resolution analysis for raw image sensor data compression," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2806–2817, Jun. 2018.
[32] T. Suzuki, "Wavelet-based spectral–spatial transforms for CFA-sampled raw camera image compression," *IEEE Trans. Image Process.*, vol. 29, pp. 433–444, 2020.
[33] M. Hernandez-Cabronero, J. Serra-Sagrista, M. W. Marcellin, and I. Blanes "Lossless compression of color filter array mosaic images with visualization via JPEG 2000," *IEEE Trans. Multimedia*, vol. 20, no. 2, pp. 257–270, Feb. 2018.
[34] Y. Lee and K. Hirakawa, "Shift- and-decorrelate lifting: CAMRA for lossless intra frame CFA video compression," *IEEE Signal Process. Lett.*, vol. 27, pp. 461–465, 2020.
[35] N. Zhang and X. Wu, "Lossless compression of color mosaic images," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1379–1388, Jun. 2006.
[36] A. Bazhyna and K. Egiazarian, "Lossless and near lossless compression of real color filter array data," *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1492–1500, Nov. 2008.
[37] S. Kim and N. I. Cho, "Lossless compression of color filter array images by hierarchical prediction and context modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 1040–1046, Jun. 2014.
[38] K.-H. Chung and Y.-H. Chan, "A lossless compression scheme for Bayer color filter array images," *IEEE Trans. Image Process.*, vol. 17, no. 2, pp. 134–144, Feb. 2008.
[39] T. Richter and S. Fossel, "Bayer pattern compression with JPEG XS," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 3177–3181.
[40] S. K. Mohammed, K. M. M. Rahman, and K. A. Wahid, "Lossless compression in Bayer color filter array for capsule endoscopy," *IEEE Access*, vol. 5, pp. 13823–13834, 2017.
[41] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1027–1034.
[42] B. E. Bayer, "Color imaging array," Granted Patent US 3 971 065 A, Jul. 20, 1976.
[43] I. Tabus and P. Astola, "Sparse prediction for compression of stereo color images conditional on constant disparity patches," in *Proc. 3DTV-Conf., True Vis.-Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jul. 2014, pp. 1–4.
[44] Y. M. Lu, M. Karzand, and M. Vetterli, "Demosaicking by alternating projections: Theory and fast one-step implementation," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2085–2098, Aug. 2010.
[45] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, "A dataset and evaluation methodology for depth estimation on 4D light fields," in *Computer Vision—ACCV 2016* (Lecture Notes in Computer Science), vol. 10113, S. H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham, Switzerland: Springer, 2016, pp. 19–34.
[46] *JPEG Pleno Database: EPFL Light-Field Data Set*. Accessed: Jan. 11, 2021. [Online]. Available: http://plenodb.jpeg.org/lf/epfl
[47] C. Shin, H.-G. Jeon, Y. Yoon, I. S. Kweon, and S. J. Kim, "EPINET: A fully-convolutional neural network using epipolar geometry for depth from light field images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4748–4757.
[48] E. C. Kaya and I. Tabus, "Corner view disparity estimation for lossless light field compression," in *Proc. 1st Eur. Light Field Imag. Workshop (ELFI)*, 2019, pp. 1–4. [Online]. Available: https://www.eurasip.org/Proceedings/Ext/ELFI_2019/Proceedings.html
[49] I. Tabus and E. Palma. *Software for Lossless Compression of Plenoptic Camera Sensor Images*. Accessed: Jan. 11, 2021. [Online]. Available: https://homepages.tuni.?/ioan.tabus/Access/SRRC_SW_TO_WEB_JAN/SRRC_SoftwareManual_Jan.pdf
[50] I. Tabus, I. Schiopu, and J. Astola, "Context coding of depth map images under the piecewise-constant image model representation," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4195–4210, Nov. 2013.
[51] 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, doi: 10.1145/214762.214771.
[52] J. Sneyers and P. Wuille, "FLIF: Free lossless image format based on maniac compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 66–70.
[53] D. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Boston, MA, USA: Kluwer, 2002.
[54] *Kakadu Software*. Accessed: Jan. 11, 2021. [Online]. Available: https://kakadusoftware.com
[55] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

IOAN TABUS (Senior Member, IEEE) received the Ph.D. degree (Hons.) from the Tampere University of Technology, Finland, in 1995. He held teaching positions at the Department of Control and Computers, Politehnica University of Bucharest, from 1984 to 1995. Since 1996, he has been a Senior Researcher, and since January 2000 a Professor with the Department of Signal Processing, Tampere University of Technology, which was merged into the Tampere University, in 2019. He is the coauthor of two books and more than 250 publications in the fields of signal compression, image processing, bioinformatics, and system identification. His research interests include light field image processing, plenoptic image compression, point clouds compression, audio, image, data compression, genomic signal processing, and statistical signal processing. He was a co-recipient of 1991 Train Vuia Award of Romania, the 2001 NSIP Best Paper Award, the 2004 NORSIG Best Paper Award, the 2016 3DTV Best Paper Award, and the ICIP 2017 Light Field Image Coding Challenge Award. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and *Signal Processing* (EURASIP). He has served as a Guest Editor for special issues for the *IEEE Signal Processing Magazine*, *Signal Processing* (EURASIP), and the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING. He was the Editor-in-Chief of the *EURASIP Journal on Bioinformatics and Systems Biology*, from 2006 to 2014.

EMANUELE PALMA (Student Member, IEEE) was born in Rome, Italy, in 1992. He received the B.S. and M.S. degrees in electronic engineering (BS) and information and communication technologies engineering from the University of Roma Tre, Roma, Italy, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Computing Sciences Unit, Tampere University, under the supervision of Prof. I. Tabus.

He is currently an Early Stage Researcher of the H2020 Marie Skłodowska-Curie project ImmerSAFE: Immersive Visual Technologies for Safety Critical Applications. His research interests include light field processing, light field compression, point clouds compression, image and data compression, multimedia signal processing, and multimedia communications.