LOSSY COMPRESSION OF LENSLET IMAGES FROM PLENOPTIC CAMERAS COMBINING SPARSE PREDICTIVE CODING AND JPEG 2000

Ioan Tabus, Petri Helin, Pekka Astola

Tampere University of Technology
Laboratory of Signal Processing
P.O. Box 553, FI-33101, Tampere, Finland

ABSTRACT

This paper proposes a lenslet image compression method scalable from low bitrates to fully lossless. The subaperture images are split into two sets: a set of reference views, encoded by a standard lossy or lossless compressor, and the set of dependent views, which are reconstructed by sparse prediction from the reference set using the geometrical information from the depth map. The set of reference views may contain all views and all views may also be dependent views, in which case the sparse predictive stage does not reconstruct from scratch the views, but it refines in a sequential order all views by combining in an optimal way the information about the same region existing in neighbor views. The encoder transmits to the decoder a segmented version of the scene depthmap, the encoded versions of the reference views, displacements for each region from the central view to each of the dependent views, and finally the sparse predictors for each region and each dependent view. The scheme can be configured to ensure random access to the dependent views, while the reference views are compressed in a backward compatible way, e.g., using JPEG 2000. The experimental results show performance better than that of the baseline standard compressor used, JPEG 2000.

***Index Terms*—** JPEG-PLENO, lenslet image compression, sparse prediction, JPEG 2000, depthmap compression, image warping

1. INTRODUCTION

The paper introduces a scalable lossy-to-lossless compression of lenslet plenoptic images, as a proposal in response to the Call for Proposals (CfP) of the JPEG committee [1][2]. The content is conceptually simple: we try to complement a very efficient and flexible image compression standard, in essence JPEG 2000, by adding features for exploiting the similarites existing between the views extracted from the lenslet image.

The proposed scheme consists in modelling the geometry of the scene and the geometry of the lenslet array and providing by these models necessary information for producing warping of a view into another view, using flexible interpolators implemented by sparse predictors. The geometry information is encoded first, so that the decoder can operate in synchrony when performing warping operations over the regions of the segmentation (each view has its own segmentation). The encoding of the lenslet image consists in selecting groups of reference views, to be encoded by JPEG 2000, while the rest of the views, called here dependent views, are encoded predictively from one of the reference views. An initial version of the whole lenslet image (and hence of all the views) can be encoded first with the JPEG 2000 coder, at a fraction of available bitrates, in which case the predictive stage refines a region in the current view, utilizing the highly similar versions of the region in the neighbor views.

The lossy encoding of lenslet images was already the subject of one grand challenge at ICME in 2016, in which five papers where published [5]-[9], with the objective and subjective results compared in [10]. The presented solutions either operated on the lenslet image as a whole, or first decomposed the lenslet into a stack of views, after which the stack of views was encoded as a pseudo video sequence by a video encoder, which exploited the similarity between closeby views. Our paper takes an approach belonging to the second group, except that it is not using a video coding algorithm, but instead it handles in an explicit way the warping of the regions of the scene, and creates flexible interpolators using sparse predictors. In this paper we present solutions for the encoding and decoding stages, assuming implicitely that the render is the reference render used to generate the reference images in the CfP.

The present paper is organized as follows. In Section 2 we present the block scheme and the main functional blocks. In Section 3 we present the experimental results and finally we conclude in Section 4.

2. THE PROPOSED COMPRESSION SCHEME

2.1. The main functional blocks used in the algorithm

Figure 1 illustrates the proposed encoding scheme as a block diagram. Next, we describe the functional blocks.

Block 1: Conversion of the lenslet image to a stack of subaperture images: We are given the lenslet image $\mathbf{\Lambda}$ having three color components on a grid of size $(m_r \times m_c) = (5368 \times 7728)$, where the element $\Lambda(x, y, c)$ is indexed by the row index x and the column index y (hence horizontal axis is Oy and vertical axis Ox) and the color component index $c \in \{0, 1, 2\}$ stands for the RGB color components $\{0, 1, 2\} \equiv \{R, G, B\}$. This image may be converted to a different color space prior to encoding, and converted back to the RGB space after decoding.

An interesting color transformation is rgb-to-ycrcb, and for the case of lossless compression we found that transforming to a single component mosaic image prior to encoding and then recovering an rgb image by demosaicng after decoding provides good results.

We use a template function to mark the pixels within a superpixel, as shown in Figure 2 a). A view collects each pixel with a given index γ from each superpixel, resulting in a rectangular array of size $(n_r \times n_c)$. The lenslet image can also be represented as a collection of views $\mathcal{I} = \{L_\gamma, \gamma \in \Gamma_{165}\}$, with each view a color image $L_\gamma : \{1, \ldots, n_r\} \times \{1, \ldots, n_c\} \times \{0, 1, 2\} \to \{0, \ldots, 2^{10} - 1\}$, with $L_\gamma(i, j, c)$ indexed by i, j for row and column in subaperture

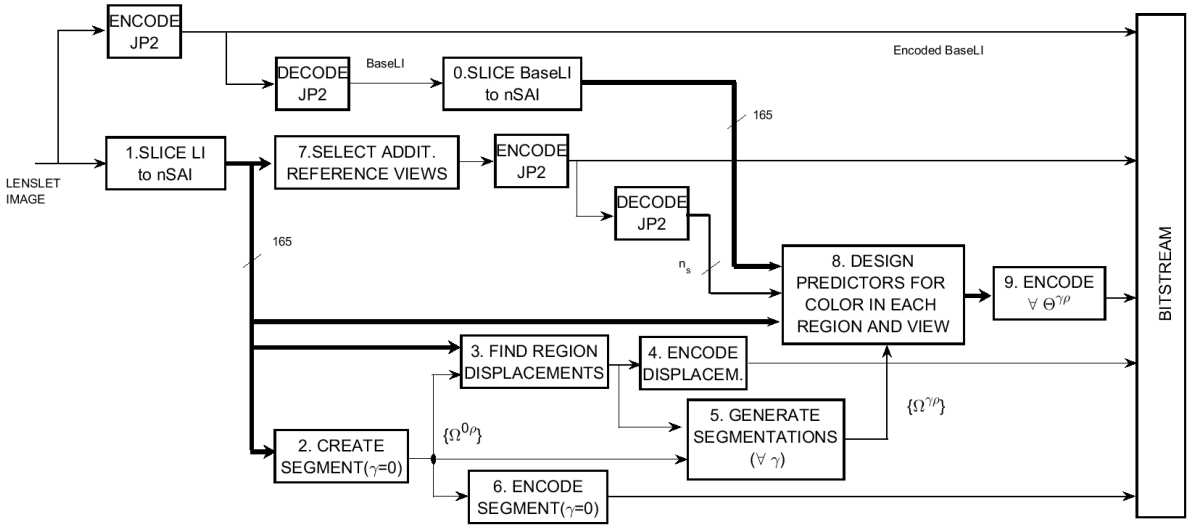


Fig. 1. Block diagram of the encoder. 1)The input lenslet image LI is converted to a LF structure of 165 non-rectified sub-aperture images *nSAI*. 2) The segmentation routine finds the segmentation of the central SAI. 3) Each region from central view segmentation is warped to each of the side views by displacements estimated in this module. 4) The found displacements are encoded. 5) The warped regions are assembled in 165 segmentations, one for each view. 6) The segmentation of the central view is encoded in the bitstream. 7) The additional reference views are selected and encoded to the bitstream. 8) For each view and each region, a sparse predictor is designed, using as regressors the background JP2 decoded image and the additional reference images. 9) The sparse predictor structure and predictor coefficients are encoded. The block numbers refer to the paragraph in text where they are described.

image and c for the color component. Note that the considered sub-aperture images are resulting directly from $\mathbf{\Lambda}$ and are different of the subaperture images *after* the processing chain of rectification. The same template function shown in Figure 2 a) gives the indexing of the views in the LF structure of non-rectified sub-aperture images.

In the system of coordinates of the sensor the hexagonal lattice has one axis slightly rotated with an angle α from the horizontal axis, and there is also horizontal shift Δ_y and vertical shift Δ_x of the lattice nodes. The location of the microlense centers in the integer coordinates (x, y) of the sensor are functions $x_C(i, j, \mathbf{\Psi}), y_C(i, j, \mathbf{\Psi})$ computed by the function *LFBuildHexGrid* from the lightfield toolbox [11], which needs as input the parameters $\mathbf{\Psi} = \{\alpha, \Delta_x, \Delta_y\}$ stored in the variables *LensletGridModel*. The mapping from the lenslet image to the view image is $L_\gamma(i, j, c) = \Lambda(x_C(i, j, \mathbf{\Psi}) + k_\gamma, y_C(i, j, \mathbf{\Psi}) + \ell_\gamma, c)$, where the template functions $\gamma \to (k_\gamma, \ell_\gamma)$ and $(k, l) \to \gamma_{k,\ell}$ can be seen from Figure 1. The conversions from one representation to the other

$$L_\gamma(i(x, y), j(x, y), c) = \Lambda(x, y, c) \quad (1)$$

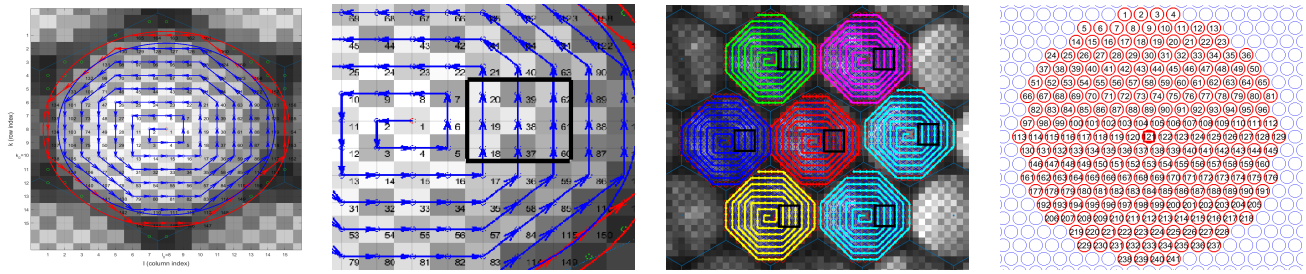
$$\Lambda(x_C(i, j, \mathbf{\Psi}) + k_\gamma, y_C(i, j, \mathbf{\Psi}) + \ell_\gamma, c) = L_\gamma(i, j, c) \quad (2)$$

can be computed efficiently. The direct conversion is used at the encoder and the inverse conversion at the decoder.

Block 2: Estimating and quantizing the disparity map: In this paper we use the estimation algorithm described in [12]. The depthmap provided by the algorithm has a high resolution, in practice almost 100 different levels. The regions having the same depth levels are the fundamental elements in the predictive encoding scheme, since we design a distinct optimal predictor for each region, and for each view. For each such predictor we will have to encode the structure and the coefficients of the predictor, and hence the bitrate for predictors is proportional to the number of levels in the disparity image. We notice that a good performance can be obtained uniformly over whole rate range if we re-quantize the depth values to a number of $N_\rho = 16$ levels. Further tuning of this parameter can improve the performance, and hence N_ρ could be left as an important configuration parameter. We note finally that the image is segmented in N_ρ segments, the pixels in each segment having the same depth. There are in average $n_r \times n_c/N_\rho$ pixels in a region and we note that a region is formed of many connected componets, possibly spread all over the scene.

Block 3: Finding the displacements of a region from the central view to a side view: The displacements of the regions are estimated and transmitted as described in our lossless compression paper [15] to which we send for a detailed description. However, a notable difference appears because the set of possible displacements is defined by the hexagonal lattice structure of the grid. The possible set of directions is marked in Figure 2 d) and consists of 241 vectors. Hence one region can be moved from the central view to a side view by using one of the 241 vectors (we found that in lenslets the maximum disparity of a pixel from the center view to the side view is about 8 pixels). We take each region in turn and find the best vector of displacements as the one giving the smallest MSE between the color of the pixels of the region in the center view and the color of pixels in translated region in the side view. The problem of finding the best displacement becomes that of finding the best sparse predictor of order 1, which has as desired value $L_\gamma(i_r, i_c, c)$, collected from a side view at location (i_r, i_c), when the template of prediction of Figure 2 d) is applied centered on the pixel (i_r, i_c) on the central view. The optimal prediction model will emulate a prediction of the form $\hat{L}_\gamma(i_r, i_c, c) = \theta_1 L_\gamma(i_r - i_{rd}, i_c - i_{cd}, c)$, but where the horizontal and vertical direction displacements (i_{rd}, i_{cd}) are represented (and encoded) as the index in the template of Figure 2 d).

Block 4: Encoding the displacements of a region from the central view to a side view: In principle the displacement vector needs about 8 bits per vector (the alphabet size is 241). The template of



(a) (b) (c) (d)

Fig. 2. (a) Spiral scanning of a microlens block. The template $\boldsymbol{\gamma}$ allocates an index $\gamma(k,l) \in \{1,\dots 165\}$ to the pixel at (k,l), e.g. $\gamma(8,8) = 1$ and $\gamma(8,7) = 2$. The template selects 165 pixels that have the centers within the octogone inscribed in the circle of radius 8.25 pixels. The views are marked in the same manner by the vector γ; (b) and (c) Prediction template for Sparse Predictors: one predictor has 63 possible regressors, taken from nine neighbor views and from seven superpixels. (b) The nine neighbor views involved, seen in the lenslet sensor grid (the nine views in the bold enclosed black square, including the views $\gamma = 20, 39, 62, 19, 38, 61, 18, 37, 60$. (c) The seven superpixels involved, marked by different colored spirals, and inside each spiral the nine views involved. In total there are 63 marked pixels. (d) Displacement template containing 241 pixels on the hexagonal grid of one non-rectified SAI image. Each template location defines a possible displacement vector for a region, i.e 123 signifies horizontal translation by two pixels to right.

the predictor covers all expected displacements of the center view to the side views, which are needed in here. One has to encode one displacement vector for each view and each region. However, we found that arranging the symbol representing each vector in a $165 \times N_\rho$ matrix results in an image which looks very similar to a disparity map (has many large constant regions). Hence we encode this $165 \times N_\rho$ "disparity" image with the CERV algorithm, which anyway is part of our routines, since it was needed in the transmission of the depth map of the central view, and obtain a remarcably low necesary bitrate.

Block 5: The warped regions are assembled into 165 segmentations: Considering a particular side view, each region is displaced from the central view with the optimally found displacement vector, which will result in a partition of this side view in regions carrying the label of the segmentation in the central view (formed by the quantized depth-map). Warping is done taking care of the possible collision of depth values, by starting the warping from the region farthest away to the closest region However, there may be missing pixels after warping, and those cases are treated as in our lossless compression method [15].

Block 6: Encoding the depth map for the central view: In here we use CERV [13] for encoding the segmentation of the depthmap, although alternatively one can use other more common encoders, like the standardized JPEG-LS [14], accepting a small loss in efficiency.

Block 7: The additional reference views are selected and encoded to the bitstream: A number of selected reference views can be encoded with JPEG 2000 at a higher PSNR than the rest of the views (the rest of the views may not be encoded at all by JPEG 2000). This is an important feature ensuring random access to the views. The most efficient current solution is to select $m_1 \times m_2$ reference views and extract a $(434m_1) \times (541m_2)$ image from the lenslet image. The obtained image is encoded with JPEG 2000. This is more efficient than encoding each reference view separately. For low bitrates one can encode with JPEG 2000 only these references and reconstruct predictively the rest of the views, using the sparse prediction of Block 8.

Block 8. Reconstructing or refining predictively a side view from available neighbor views Each region Ω_ℓ of the segmentation of a view has its own sparse predictor $\boldsymbol{\Theta}_\ell$. In the implemented program, the sparse predictor takes its inputs (its regressors) from the available reconstructions of the current view and of the 8 neighbor views, as presented in Figure 2 (a-c). The figure and its caption give close details of the linear regression model setting. The sparse predictor operates independently over the color planes, and uses the same set of coefficients for all color planes, designed to be optimal over all color planes. In the simplest form, the predictor has a single active template pixel, which will be equivalent with a simple translation. Higher order predictors perform better interpolations. For the greedy prediction Fast OLS used in this paper [16], we have available for each region Ω_ℓ a set of predictors, $\boldsymbol{\Theta}_\ell^{[1]}, \boldsymbol{\Theta}_\ell^{[2]} \dots, \boldsymbol{\Theta}_\ell^{[K]}$, indexed by their number of coefficients, $k = 1, 2, \dots, K$. In the experimental section we show results when confining all predictors to the same order, k, set according to the target bitrate. The sparse predictor design was described for a similar situation in [15], to which we send for more details.

Block 9. Encoding the predictor parameters: The sparse predictor has k nonzero parameters. The mask of nonzero parameters is encoded by transmitting the indices of the pixels from the prediction template, selected to be nonzero. The locations are sent in their oder of selection by the FastOLS algorithm. The prediction mask has 63 locations, and each location is written raw using 8 bits. Hence the cost of all prediction masks is $165 \times n_\Omega \times k \times 8bits$. The nonzero parameters are grouped according to their rank of selection by the greedy FastOLS algorithm. The first selected parameter tends to have high values, while the last selected parameter, of rank k, tends to have smaller values. For this reason the parameters are encoded in groups, one group for each rank. Every nonzero parameter θ is first quantized to a given number of bits in the fractional part, in our experiments $n_B = 12$, and then the integer value $\lfloor \theta 2^{n_B} \rceil$ is stored in a string of integers encoded using Golomb-Rice (GR) coding and the sign is sent as a raw bit. There are k groups (according to the rank of selection) and each group has its own GR parameter.

2.2. Advantages of the proposed method

Random access: The current implementation treats the case when a lenslet image is encoded. All the modules can be rewritten for the simpler case when the rectified lightfield structure LF has to be encoded directly. In that case the views may be needed to be accessed

img.	method	B_1	B_2	B_3	B_4
I_01	prop.	29.73 (0.741)	31.93 (0.816)	33.62 (0.893)	38.31 (0.964)
I_02	prop.	26.90 (0.671)	28.50 (0.770)	30.52 (0.889)	35.05 (0.964)
I_04	prop.	31.77 (0.763)	32.82 (0.843)	35.19 (0.925)	38.43 (0.973)
I_09	prop.	28.31 (0.718)	30.48 (0.804)	31.79 (0.864)	35.50 (0.952)
I_10	prop.	32.95 (0.853)	36.17 (0.907)	38.12 (0.948)	41.49 (0.977)

Table 1. Mean $PSNR_{YUV}$ as given by the reference software `difftest_ng` and in parenthesis the mean $SSIM_{YUV}$ given by `ssimdiff`.

with random access. Our encoding scheme is well suitable for this feature, e.g. by decoding first a small part (the depth map and the displacements relevant to the side view γ), then decoding the reference γ_0 (or the group of references containing it) and finally the depedent side view.

Included depthmap: The bitstream includes the depthmap, which is obtained with high precision at the encoder from the plenoptic camera using the original lenslet image. The decoder receives at no additional cost the depthmap, which may be used in many additional applications, e.g. for creating VR content.

3. EXPERIMENTAL RESULTS

The evaluation of the compression scheme is done according to the methodology established in the JPEG Pleno Call for proposals on Light Field Coding.

The image to be encoded, $\mathbf{\Lambda}$, is the RGB lenslet image of size $(n_r \times n_c) = (5368 \times 7728)$, obtained after devignetting and demosaicing, distributed by the organizers of the challenge. There are five scenes, denoted *I_01, I_02,I_04,I_09,I_10*, introduced in [17]. The presented encoding algorithm is used for encoding $\mathbf{\Lambda}$ into a bitstream having $n_r n_c \bar{B}$ bits, where $\bar{B}$ is the reported compression, in bits per pixel. The decoding algorithm produces an approximate reconstruction, denoted $\hat{\mathbf{\Lambda}}$.

The pipeline for creating from $\mathbf{\Lambda}$ a rectified lightfied structure LF is denoted $LF = \mathcal{T}(\mathbf{\Lambda})$, where LF is the rectified 15×15 array of subaperture images (SAIs) to be displayed and subjectively evaluated, and the transformation $\mathcal{T}$ is realized by the function created by the organizers using the Lightfield Toolbox [11].

3.1. Lossy coding

Each SAI in the LF is an RGB image, and the objective measures indicated by the organizers are PSNR and SSIM values, computed for each SAI (by transforming each RGB SAI to YUV colorspace and then computing weighted sum over the Y,U,V components and over all SAIs). The computation of PSNR is done using the program `difftest_ng` provided by the organizers using the parameters `--toycbcr --yuvpsnr` and similarly the SSIM is computed with `ssimdiff --nowav --lin`.

We report in Table 1 the $PSNR_{YUV}$ and $SSIM_{YUV}$ indicators for all the scenes, at all the required bitrates $B_1 = 0.005$, $B_2 = 0.02$, $B_3 = 0.1$ and $B_4 = 0.75$ bpp. In the experimental results we fixed the order of predictors depending on the available bitrate, as follows: $k = 0$ at rate B_1, $k = 2$ at rate B_2, $k = 6$ at rate B_3 and $k = 6$ at rate B_4i. A similar lenslet image encoding-decoding scheme to our proposed method can be achieved using e.g. plain JPEG 2000 or HEVC in its intra mode. They are given as reference in Table 2.

We note that the effective length of the bitsream created by our program is always conservatively taken below the target rate, e.g., at the target $B_1 = 0.005$ bpp, for image I_{01} we have used $\bar{B}_1 = 0.0049$ bpp. For the HEVC, the bitrate was chosen such that it was above the target bitrate.

img.	method	B_1	B_2	B_3	B_4
I_01	JP2	0.00%	1.74%	0.73%	0.29%
	HEVC	13.15%	11.25%	3.42%	2.75%
I_02	JP2	0.00%	0.90%	1.10%	0.65%
	HEVC	14.79%	9.95%	4.26%	2.71%
I_04	JP2	0.00%	0.00%	0.68%	0.24%
	HEVC	21.76%	12.40%	3.54%	1.24%
I_09	JP2	0.00%	2.61%	1.54%	0.58%
	HEVC	9.05%	14.00%	5.09%	2.11%
I_10	JP2	0.00%	0.64%	0.40%	0.25%
	HEVC	22.51%	9.34%	3.34%	2.01%

Table 2. The relative improvement in $PSNR_{YUV}$ compared to the HEVC-intra and the background method JPEG 2000.

Image	*I_01*	*I_02*	*I_04*	*I_09*	*I_10*
Bmax[*bpp*]	9.88	9.67	9.72	10.26	9.31
JP2 Bmax[*bpp*]	16.07	15.26	15.53	17.25	15.67

Table 3. Bitrates for lossless coding.

3.2. Lossless coding

A direct lossless encoding of the RGB image $\mathbf{\Lambda}$ by JPEG 2000 will produce a file with about 16 bpp as shown in the second row in Table 3. A much more efficient way is to perform the transformation of the RGB image into a single component mosaic image, Y_m, extracted from the RGB color planes according to the 'grbg' Bayer pattern of the camera: $Y_m(2i, 2j) = \mathbf{\Lambda}(2i, 2j, 2), Y_m(2i + 1, 2j + 1) = \mathbf{\Lambda}(2i+1, 2j+1, 2), Y_m(2i+1, 2j) = \mathbf{\Lambda}(2i+1, 2j, 1), Y_m(2i, 2j+1) = \mathbf{\Lambda}(2i, 2j + 1, 3)$.

After lossless compression of the mosaic image Y_m using lossless mode of JPEG 2000, and then performing demosaicing with the filter from [19], one gets the RGB $\hat{\mathbf{\Lambda}}$, which is a near-lossless version of $\mathbf{\Lambda}$, (e.g., for image I_01, $PSNR = 65.3$ and the needed bitrate is 6.56 bpp). Performing an additional lossless encoding of the almost null RGB image $\mathbf{\Lambda} - \hat{\mathbf{\Lambda}}$ by using lossless mode of JPEG 2000, one gets a perfect lossless reconstruction of $\mathbf{\Lambda}$. The obtained results are shown in the first row of Table 3.

4. CONCLUSIONS

The proposed algorithm has a great flexibility, using relatively simple functional blocks, in a structure which can be configured by the user. Further optimization of this structure will allow to obtain better results.

5. REFERENCES

[1] T. Ebrahimi, S. Foessel, F. Pereira, P. Schelkens, JPEG Pleno: Toward an efficient representation of visual reality, IEEE Multimedia, Vol. 23, No. 4, pp. 14–20, 2016.

[2] ISO/IEC JTC 1/SC29/WG1 JPEG, "JPEG Pleno Call for Proposals on Light Field Coding," Doc. N74014, Geneva, Switzerland, January 2017.

[3] D. Taubman and M. W. Marcellin, *JPEG 2000: Image compression fundamentals, standards and practice*. Boston, MA: Kluwer, 2002.

[4] A. Skodras, C. Christopoulos, T. Ebrahimi, "The JPEG 2000 still image compression standard", IEEE Signal Process. Mag., vol. 18, no. 5, pp. 36–58, Sep. 2001.

[5] C. Conti, P. Nunes, and L. D. Soares, HEVC-based light field image coding with bi-predicted self-similarity compensation, in 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), July 2016, pp. 1–4.

[6] 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 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), July 2016, pp. 1–4.

[7] Y. Li, R. Olsson, and M. Sjostrom, Compression of unfocused plenoptic images using a displacement intra prediction, in 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), July 2016, pp. 1–4.

[8] C. Perra and P. Assuncao, High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement, in 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), July 2016, pp. 1–4.

[9] D. Liu, L. Wang, L. Li, Z. Xiong, F. Wu, and W. Zeng, Pseudo-sequence-based light field image compression, in 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), July 2016, pp. 1–4.

[10] I. Viola, M. Rerabek, T. Bruylants, P. Schelkens, F. Pereira and T. Ebrahimi. "Objective and subjective evaluation of light field image compression algorithms." 32nd Picture Coding Symposium, Nuremberg, Germany, 2016.

[11] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," in *CVPR*, June 2013, pp. 1027–1034.

[12] T.-C. Wang, A. Efros, and R. Ramamoorthi. "Occlusion-aware depth estimation using light-field cameras," In Proceedings of International Conference on Computer Vision (ICCV), 2015.

[13] 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.

[14] M. Weinberger, G. Seroussi, and G. Sapiro, "The lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, pp. 1309–1324, August 2000.

[15] P. Helin, P. Astola, B. Rao, and I. Tabus, "Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression," in *3DTV-CON*, July 2016.

[16] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1713–1715, July 1995.

[17] M. Rerabek and T. Ebrahimi, "New light field image dataset," in 8th International Conference on Quality of Multimedia Experience (QoMEX), 2016.

[18] I. Tabus and P. Helin, "Microlens image sparse modelling for lossless compression of plenoptic camera sensor images," In Proceedings of *EUSIPCO*, Kos, August 2017.

[19] H. S. Malvar, Li-wei He and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. iii-485-8 vol.3, 2004.