Light Field Coding Using Weighted Binary Images

Koji KOMATSU^{†a)}, *Member*, Kohei ISECHI^{†b)}, *Student Member*, Keita TAKAHASHI^{†c)}, *Member*,
and Toshiaki FUJII^{†d)}, *Fellow*

SUMMARY We propose an efficient coding scheme for a dense light field, i.e., a set of multi-viewpoint images taken with very small viewpoint intervals. The key idea behind our proposal is that a light field is represented using only weighted binary images, where several binary images and corresponding weight values are chosen so as to optimally approximate the light field. The proposed coding scheme is completely different from those of modern image/video coding standards that involve more complex procedures such as intra/inter-frame prediction and transforms. One advantage of our method is the extreme simplicity of the decoding process, which will lead to a faster and less power-hungry decoder than those of the standard codecs. Another useful aspect of our proposal is that our coding method can be made scalable, where the accuracy of the decoded light field is improved in a progressive manner as we use more encoded information. Thanks to the divide-and-conquer strategy adopted for the scalable coding, we can also substantially reduce the computational complexity of the encoding process. Although our method is still in the early research phase, experimental results demonstrated that it achieves reasonable rate-distortion performances compared with those of the standard video codecs.

key words: light-field, scalable coding

1. Introduction

A light field, which is equivalent to a set of multi-view images captured with both horizontal- and vertical-viewpoint displacements, has many applications such as depth estimation [1], [2], free-viewpoint video [3]–[5], and 3-D display [6]–[8]. Thanks to the introduction of light field cameras (e.g., Lytro Illum) [9]–[13], dense light fields, which have dozens to hundreds of images with very small viewpoint intervals, have recently attracted much research interest [2], [7], [14]–[16].

One of the important research issues is efficient coding (compression) schemes for such dense light field data, because the data amount increases as the number of images increases. A straightforward approach to this problem is to simply adopt well-established image/video coding technologies including inter/intra-frame prediction, discrete cosine transform (DCT)/discrete wavelet transform, etc. Using standard video codecs such as H.265 [17] with relatively

small modifications have been proven to yield good coding performance for light field or multi-view coding [18]–[23]. However, we believe that these *generic* technologies are not necessarily optimal for dense light fields, because dense light fields have greater redundancies among the viewpoint compared to general video/multi-view sequences, for which such codecs have been optimized. Therefore, there should be coding schemes other than the standard codecs that are advantageous specifically for dense light fields.

With this in mind, we propose an efficient coding scheme for a dense light field*. The key idea behind our scheme is that a light field is represented using only weighted binary images, where several binary images and corresponding weight values are chosen so as to optimally approximate the light field. Therefore, only those binary images and weight values are to be saved to reconstruct the original light field later. This idea was originally developed to generate temporal binary sequences for multi-view displays with active shutter glasses [25]. However, to the best of our knowledge, we are the first to validate that this idea is suitable for light field coding.

Our coding scheme is completely different from those of modern image/video coding standards. One advantage of our method is the extreme simplicity of the decoding process, which will lead to a faster and less power-hungry decoder than those of the standard codecs. Another useful aspect is that our coding method can be made scalable, where the accuracy of the decoded light field is improved in a progressive manner as we use more encoded information, in a manner similar to that of quality scalability [26]. Thanks to the divide-and-conquer strategy adopted for the scalable coding, we can also substantially reduce the computational complexity of the encoding process. Although our method is still in the early research phase, experimental results demonstrated that it achieves reasonable rate-distortion performances compared with those of the standard video codecs.

2. Proposed Method

2.1 Coding Light Field with Weighted Binary Images

Assume that a 4-D light field to be compressed, $L(s, t, x, y)$,

*This paper is an extension of our conference paper [24]. We included in this paper more thorough descriptions on our method and several new experimental results.

Manuscript received January 25, 2019.

Manuscript revised May 20, 2019.

Manuscript publicized July 3, 2019.

[†]The authors are with Department of Information and Communication Engineering, Graduate School of Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.

a) E-mail: komatsu@fujii.nuee.nagoya-u.ac.jp

b) E-mail: isechi@fujii.nuee.nagoya-u.ac.jp

c) E-mail: keita.takahashi@nagoya-u.jp

d) E-mail: t.fujii@nagoya-u.jp

DOI: 10.1587/transinf.2019PCP0001

Fig. 1 Light field representation using binary images $B_n(x, y)$ and corresponding weights $r_n(s, t)$.

is given as a set of multi-view images. The viewpoints, which are arranged in a 2-D grid, are specified as (s, t) ($s = 1, \dots, S$, $t = 1, \dots, T$), and the pixels are indicated as (x, y) ($x = 1, \dots, X$, $y = 1, \dots, Y$). We first assume that $L(s, t, x, y)$ takes a gray-level value, and is later extended to an RGB color value.

The key concept behind our scheme is that the target light field $L(s, t, x, y)$ can be approximated using only N binary images $B_n(x, y) \in \{0, 1\}$ ($n = 1, \dots, N$) and corresponding weights $r_n(s, t) \in \mathcal{R}$. As illustrated in Fig. 1, $L(s, t, x, y)$ is written in a sum-of-product form as

$$L(s, t, x, y) \approx \sum_{n=1}^N B_n(x, y) r_n(s, t). \quad (1)$$

Note that binary images $B_n(x, y)$ are common to all viewpoints, but weights $r_n(s, t)$ can differ from viewpoint to viewpoint. In other words, a set of binary *basis images* is selected to capture a common structure among all viewpoints, and the differences among the viewpoints are represented with pixel-independent weights $r_n(s, t)$. We propose to use N sets of $B_n(x, y)$ and $r_n(s, t)$ as a compressed representation of the original light field $L(s, t, x, y)$.

Encoding. To obtain a compressive representation from the given light field, we solve an optimization problem described as

$$\arg \min_{B_n(x, y), r_n(s, t)} \sum_{s, t, x, y} \left| L(s, t, x, y) - \sum_{n=1}^N B_n(x, y) r_n(s, t) \right|^2. \quad (2)$$

We have two sets of unknowns, $B_n(x, y)$ and $r_n(s, t)$, for which we take an alternative approach. Specifically, we first initialize the binary images $B_n(x, y)$ and repeat the two following steps until convergence. We do not need to initialize the weights $r_n(s, t)$. (i) We fix the binary images $B_n(x, y)$ and optimize the weights $r_n(s, t)$. This step falls into the category of solving the standard least squares problems, for which many good solvers are available. We used the “solve” function in OpenCV as a numerical solver for this optimization. (ii) We fix the weights $r_n(s, t)$ and optimize the binary

images $B_n(x, y)$. This problem can be solved individually for each pixel (x, y) , but it is a binary combinatorial optimization, which is known to be NP-hard. We simply use brute-force search for this problem, but a substantial increase in speed can be achieved with a scalable coding framework, as described in Sect. 2.2.

In our current implementation, all $B_n(x, y)$ are initialized to the same image that is obtained by applying binary thresholding to the top-leftmost image of the target light field. Since all the binary images are identical at this point, $r_n(s, t)$ cannot be determined uniquely in the optimization step (i) of the first iteration. Here, we simply use the solve function in OpenCV to obtain one of the solutions (i.e., a set of weights) in the least-squares sense, and move on to the next step. We empirically adopted this initialization scheme, but there might be better ones.

In Figs. 2 and 3, we show an example obtained with the Lego truck dataset [27], which has 17×17 views and 160×120 pixels[†] in grayscale. Figure 2 shows the binary images $B_n(x, y)$ and corresponding weights $r_n(s, t)$ obtained by solving Eq. (2) with $N = 10$. Figure 3 shows the error maps (pixel-by-pixel absolute difference) between the original and decoded multi-view images. Here, the decoded images are computed with the righthand side of Eq. (1) using the binary images and weights shown in Fig. 2. It can be seen that the error tends to be larger for the viewpoints located farther from the center and for the regions having larger disparities (e.g., the front wheel of the truck). This tendency can be attributed to the fact that we use the common binary images $B_n(x, y)$ for all the viewpoints (s, t) , which imposes a limitation on the representation capability for the original 4-D signal $L(s, t, x, y)$. Note that the error we show here is obtained with $N = 10$ for visualization purposes, but the error can be reduced by using a larger N .

Compression ratio. Similarly to other coding schemes, our scheme has a trade-off between rate (number of bits) and distortion (accuracy of decoded data). This trade-off can be controlled by simply changing the number of binary images N . If we retain N binary images $B_n(x, y)$ and weights $r_n(s, t)$ as they are (in a raw data format), the compression ratio is given by

$$\text{compression ratio} = \frac{N(XY + STb_r)}{STXYb_L}, \quad (3)$$

where b_r and b_L denote respectively the bit depths used to represent weights $r_n(s, t)$ and light field $L(s, t, x, y)$. For example, for a light field with $X = 160$, $Y = 120$, $S = 17$, $T = 17$, and $b_L = 8$, we can achieve 0.54% with $N = 10$ and $b_r = 16$. We can further reduce the data amount by applying lossless coding (e.g. gzip) to the raw data.

Decoding. An important advantage of our scheme is its extreme simplicity of decoding, as shown in Fig. 1. The light field can be reconstructed only with very simple operations, without motion (disparity in this context) compensated prediction, intra-prediction, inverse discrete co-

[†]The spatial resolution is reduced from the original one.

Fig. 2 Binary images $B_n(x, y)$ and corresponding weights $r_n(s, t)$ obtained with $N = 10$.

Fig. 3 Error map between original and decoded multi-view images with $N = 10$.

sine transform, etc. This simplicity should lead to faster and less power-hungry decoder implementation for both a field-programmable gate array (FPGA) and an application specific integrated circuit (ASIC) compared to sophisticated standard video codecs.

2.2 Scalable Coding

In this section, we further extend our scheme to make it scalable, where the accuracy of the decoded light field is improved in a progressive manner as we use more encoded information. Although this scalable framework slightly decreases the coding efficiency (rate-distortion performance) compared to its non-scalable counterpart, it has a significant positive side effect; we can substantially reduce the computational complexity for encoding thanks to the divide-and-conquer strategy that is inherent to the scalable structure.

Notations. We first introduce notations. Assume that we use N binary images in total to represent the target light field. We divided them into M groups, where M corresponds to the number of levels (layers) for scalability. To be specific, we define M sets of integers as $\{N_1, \dots, N_M\}$ that satisfy $N_i \cap N_j = \emptyset$ for $i \neq j$ and $\bigcup_{m=1}^M N_m = \{1, \dots, N\}$. Symbol $|N_m|$ denotes the number of binary images for the m -th level. We have $\sum_m |N_m| = N$ by definition.

Encoding. The encoding process proceeds as the approximation accuracy improves progressively. An illustration and the specific algorithm are respectively given in Fig. 4 and Algorithm 1. For the first level, the target light field $L_1(s, t, x, y)$ is set to the original light field $L(s, t, x, y)$. At each level, we explore the best approximation for the target light field $L_m(s, t, x, y)$ using only $|N_m|$ binary images $B_n(x, y)$ and corresponding $ST \times |N_m|$ weights $r_n(s, t)$. We optimize $B_n(x, y)$ and $r_n(s, t)$ only for $n \in N_m$ as follows:

$$\arg \min_{B_n(x, y), r_n(s, t)} \sum_{s, t, x, y} \left| L_m(s, t, x, y) - \sum_{n \in N_m} B_n(x, y) r_n(s, t) \right|^2. \quad (4)$$

Then, we carry over the residual from the current level to

Fig. 4 Scalable coding.

the next level. More specifically, the target light field for the next level is given as

$$L_{m+1}(s, t, x, y) = L_m(s, t, x, y) - \sum_{n \in N_m} B_n(x, y) r_n(s, t). \quad (5)$$

Equations (4) and (5) are repeated to the M -th level. Equation (5) can be transformed as

$$L(s, t, x, y) = \sum_{m=1}^M \sum_{n \in N_m} B_n(x, y) r_n(s, t) + L_{M+1}(s, t, x, y). \quad (6)$$

The residual monotonically reduces as the level proceeds. Unless all the elements of $L_m(s, t, x, y)$ are zero,

$$\begin{aligned} & \sum_{s, t, x, y} |L_{m+1}(s, t, x, y)|^2 \\ &= \sum_{s, t, x, y} \left| L_m(s, t, x, y) - \sum_{n \in N_m} B_n(x, y) r_n(s, t) \right|^2 \\ &< \sum_{s, t, x, y} \left| L_m(s, t, x, y) - \sum_{n \in N_m} 0 \cdot 0 \right|^2 \\ &= \sum_{s, t, x, y} |L_m(s, t, x, y)|^2 \end{aligned} \quad (7)$$

is satisfied, because $B_n(x, y)$ and $r_n(s, t)$ for $n \in N_m$ are determined to minimize the squared sum of the residual in accordance with Eq. (4). This indicates that less information is left for the higher level. Therefore, $L_{M+1}(s, t, x, y)$ in Eq. (5) converges to 0 when M is sufficiently large. As a result, we obtain a scalable representation with M levels as

$$L(s, t, x, y) \simeq \sum_{m=1}^M \sum_{n \in N_m} B_n(x, y) r_n(s, t). \quad (8)$$

Accordingly, from the encoded data, we can extract a less accurate light field by using only the first M' ($M' < M$) levels. As the number of levels M' increases, the accuracy of the decoded light field improves progressively. This functionality would be useful for many practical scenarios such as adaptive rate-control and flexible user adaptation.

Computational complexity. The scalable framework mentioned above can be understood as a *divide-and-conquer* approach to the original problem. Instead of optimizing all the N binary images $B_n(x, y)$ and corresponding weights $r_n(s, t)$ simultaneously, as in Eq. (2), the scalable framework divides them into M groups and optimizes them one by one,

Algorithm 1 Scalable light field coding using weighted binary images

Input: $L(s, t, x, y)$
Output: $B_n(x, y), r_n(s, t)$ ($n = 1, 2, \dots, N$)
Initialize $L_1(s, t, x, y) \leftarrow L(s, t, x, y)$
for $m = 1$ to M **do**
 Obtain $B_n(x, y), r_n(s, t)$ ($n \in N_m$) using Eq. (4)
 Carry over the residual using Eq. (5)
 $m \leftarrow m + 1$
end for

as in Eq. (4). Generally, a divide-and-conquer approach cannot attain the global optimum, but significant increase in speed can be achieved by reducing the search space. In our case, the coding efficiency (rate distortion performance) with the scalable framework is slightly worse than that with the non-scalable one, as will be shown later. However, this scalable framework substantially reduces the computational complexity for encoding compared to its non-scalable counterpart. Specifically, we can gain a significant increase in speed for the optimization of binary images $B_n(x, y)$, which is the main computational bottleneck of our encoding process. The computational complexity for the original problem— N -bit binary combinatorial optimization is $O(2^N)$. When it is divided into M sub-problems, it will reduce to $O(\sum_m 2^{|N_m|})$. For example, with $N = 12$, $M = 4$, and $|N_m| = 3$ for all m , the scalable framework reduces the computational cost to $1/128$.

2.3 Handling Color Channels

We present two possible approaches to handling RGB color channels in our coding scheme.

RGB domain. A straightforward approach is to apply the model of Eq. (1) individually to the three RGB color channels. In this case, the data amount triples in a raw data format compared to the case with a single channel. However, it would be better to adopt a joint representation that can exploit a strong correlation among the three channels. More specifically, we use the common binary images for all color channels, but with different weights for different colors, as

$$L^c(s, t, x, y) \simeq \sum_{n=1}^N B_n(x, y) r_n^c(s, t), \quad (9)$$

where $c \in \{0, 1, 2\}$ denotes the three RGB color components. This joint representation improves coding efficiency, as discussed later. In this case, the compression ratio is represented as

$$\text{compression ratio} = \frac{N(XY + 3STb_r)}{3STXYb_L}. \quad (10)$$

where b_L denotes the bit depth used to represent each color component of $L^c(s, t, x, y)$.

YUV domain. Another promising approach is to work on the problem in the YUV domain, as has been done in the image/video coding community. First, we convert RGB images into YUV420 format, where the resolutions for U and V channels are halved both in the horizontal and vertical directions. We encode Y channel by itself using Eq. (1), but we use the common binary images for U and V channels similarly to Eq. (9).

$$\begin{aligned} L^{y^*}(s, t, x, y) &\simeq \sum_{n=1}^{N_{y^*}} B_n^{y^*}(x, y) r_n^{y^*}(s, t) \\ L^{u^*/v^*}(s, t, x, y) &\simeq \sum_{n=1}^{N_{u^*v^*}} B_n^{u^*v^*}(x, y) r_n^{u^*/v^*}(s, t), \end{aligned} \quad (11)$$

where y^* , u^* and v^* are used to indicate the corresponding YUV channels, u^*/v^* means “ u^* or v^* ”, and N_{y^*} and $N_{u^*v^*}$ are respectively the number of binary images used for Y and UV channels. Note that the binary images $B_n^{u^*v^*}(x, y)$ are shared between the U and V channels; they are used to represent both $L^{u^*}(s, t, x, y)$ and $L^{v^*}(s, t, x, y)$. In this case, the compression ratio in a raw data format is obtained with

$$\frac{N_{y^*}(XY + STb_r) + N_{u^*v^*}(XY/4 + 2STb_r)}{3STXYb_L}. \quad (12)$$

Here, the resolution reduction in U and V channels is also included in the compression process.

3. Experiments

We implemented our method using C++ with the basic part of our software made available from our website [28]. For optimizing Eq. (2) and Eq. (4), the number of iterations was set to 20. To compress the raw data (binary patterns and weight values), gzip ver. 1.6 was adopted as an optional post-processing.

3.1 Performance of Our Method

We first demonstrate our method’s performance in different configurations, using truck dataset [27], which is shown in the second column in Fig. 8. Our scheme was first evaluated with the grayscale version.

Figure 5 shows the peak signal-to-noise ratio (PSNR) of reconstructed images using Eq. (2) with different numbers of binary images ($N = 8, 10$, and 12). Each square corresponds to each viewpoint. As N increases, the entire reconstruction accuracy increases accordingly. The PSNR values are lower for the viewpoints located at peripheral positions than those located near to the center. This tendency is consistent with the error maps shown in Fig. 3.

Next, we evaluated our method’s rate-distortion performance and encoding time with and without the scalable framework. The PSNR values were calculated from the mean squared errors over all the viewpoints and pixels. The bitrate values were calculated from the sizes of files that were compressed with gzip. For the case with the scalable framework, we used the same number of binary images, denoted as N' , for all levels. Computational times were measured on a Desktop PC running Windows 7 pro equipped with Intel (R) Core (TM) i7-4790 3.6 GHz CPU and 12.0 GB main memory. As shown in Fig. 6(a), using the scalable framework slightly decreases the coding efficiency compared to its non-scalable counterpart. Figure 6(b) shows the encoding time with and without the scalable framework, where the total number of binary images N was fixed to 12. We can see that an approximately 100-fold increase in speed is achieved with the scalable framework. In terms of the balance between the rate-distortion performance and the encoding time, $N' = 3$ is likely to be the best parameter in the scalable framework. Therefore, we used this in Sects. 3.2 and 3.3.

Fig. 5 PSNR of reconstructed images using Eq. (2) with different N .

(a) PSNR

(b) Encoding time with $N = 12$

Fig. 6 Our method's rate-distortion performance and encoding time with and without scalable framework.

We also tested our method with the RGB color version of the truck dataset. We compared three methods, which differ in how they handle color channels. The first method handles each of the RGB color channels individually applying the same representation as Eq. (1) to each of them (RGB independent method). The second method shares the binary images among the RGB color channels as represented by Eq. (9) (RGB shared method). The third method first converts the images to YUV420 format and applies the representation given by Eq. (11) (YUV method). The PSNR values were calculated from the mean squared errors over all the viewpoints, pixels, and color (RGB) channels[†]. The bitrate values were measured with gzip compression.

[†]In the image/video coding community, PSNR values are often measured in the YUV domain rather than the RGB domain. However, in this paper, we focused on reconstruction accuracy over the original RGB information, rather than the perceptual quality of the reconstructed light fields.

(a) YUV methods

(b) Comparison of three methods

Fig. 7 Rate distortion performance for color light field.

We first evaluated the rate distortion performance of YUV method alone. The number of binary images for Y channel $N_{y'}$ and for UV channels $N_{u'v'}$ were respectively set to 5–10 and 3–10. Figure 7(a) shows several rate-distortion curves, each of which was drawn with a fixed $N_{y'}$ while the value for $N_{u'v'}$ is changing. As observed from this graph, it is important to find the optimal combinations of $N_{y'}$ and $N_{u'v'}$. For example, three configurations with $(N_{y'}, N_{u'v'}) = (5, 6)$, $(6, 5)$, and $(7, 4)$ yielded almost the

Datasets	Truck	Amethyst	Bulldozer	Bunny	Crystal	Knight
# of views	17 × 17					
size	160 × 120	96 × 128	192 × 144	128 × 128	128 × 128	128 × 128
Top left image
Difference between top left and top right images

Fig. 8 Datasets.

same bitrates, but resulted in significantly different reconstruction accuracy. The red squares in this graph indicate the configurations that are seemingly near to the upper-bound performance.

Figure 7 (b) shows rate-distortion performances for the three methods. For the YUV method, only the configurations marked with red squares in Fig. 7 (a) are plotted. We can see that using correlations among the color channels contributed to the better rate-distortion performances. In particular, handling color information in YUV420 format yielded the best performance.

3.2 Comparison with Video Coding Standard

As a representative of modern video coding standards, we used H.265 (ISO/IEC 23008-2) [17], which is often called HEVC. We adopted two implementations for HEVC: FFmpeg ver. 4.1 with default parameters and the HEVC Test Model [29] with a random access configuration [30]. To ascertain the effect of inter-frame prediction, we also tested all intra mode with the HEVC Test Model. When we applied these video codecs to a light field dataset, we aligned images in the dataset in the major row order and regarded them as a video sequence.

As shown in Fig. 9, we compared the rate-distortion performances over several grayscale datasets taken from [27], the details of which are shown in Fig. 8. We also present the results obtained with the color truck dataset in Fig. 10. Our method's performance appears to depend on the image differences between the viewpoints. It achieved a good rate-distortion performance for a dataset with small differences, such as Amethyst, but could not achieve high PSNR values for a dataset including large differences, such as Bulldozer. It achieved a reasonable performance overall, comparable to that of FFmpeg but moderately inferior to that of the HEVC Test Model. We believe these results are promising, considering the fact that HEVC is the state-of-the-art video coding standard that has been optimized with a significant amount of labor and time.

Finally, we compared the decoding times between our method and HEVC codecs using the truck dataset in

grayscale. We used the same computer as the one for Fig. 6 (b), but we measured the decoding time on Ubuntu 16.04 installed on Virtual Box ver 5.1.16 due to the availability of software. We measured the user time for executing decoding processes with “time” command. We repeated the measurement 100 times and obtained the average for each method, which is plotted in Fig. 11. For our method, N was set to 10 and the outputs were written as pgm files. The time for unzip process was negligible. For FFmpeg and the HEVC Test Model, the outputs were respectively written in pgm files and YUV files. We set the parameters for these codecs so that they resulted in almost the same PSNR as that of the proposed method. It can be seen from the graph that our method runs much faster than HEVC codecs. This can be attributed to the simplicity of our method; as mentioned earlier, its decoding process is carried out with simple sum-of-product operations, while HEVC requires complex inter/intra-frame prediction and transforms.

It should be noted that the decoding times depend heavily on the implementations and computing environments. We believe that our method would benefit more from dedicated hardware implementations using a field-programmable gate array (FPGA) or an application specific integrated circuit (ASIC), because its simplicity would lead to less hardware complexity and less power consumption compared to the standard video codecs.

3.3 Discussion

One remaining issue with our method is to handle light fields that have higher spatial resolutions. To see the effect of the resolution, we arranged another experiment. We used the Lego truck dataset [27], which originally had 1280×960 pixels. We converted it into grayscale, and resized it into different resolutions (320×240 , 256×192 , and 160×120) using Lanczos interpolation. We measured the rate-distortion performance and encoding time using the Ubuntu environment mentioned earlier. As can be seen from Fig. 12, increasing the resolution results in worse rate-distortion performance both for the non-scalable and scalable coding schemes. One possible reason is that increasing the resolution leads to

Fig. 9 Comparison of R-D curves (grayscale) between our method (with gzip) and HEVC.

Table 1 Encoding time for different resolutions (in seconds).

Resolution	160×120	256×192	320×240
Proposed (non-scalable, $N = 10$)	1833	4686	7316
Proposed (non-scalable, $N = 8$)	437	1123	1752
Proposed (scalable, $N = 12$)	117	255	398
Proposed (scalable, $N = 9$)	87.7	190	302
HEVC Test Model	34.6	103	158
FFmpeg	0.82	1.94	3.21

larger disparity ranges, making the light field more difficult to be represented using common binary images. Increasing the resolution also leads to the increase of the encoding time, as shown in Table 1. Compared to the standard video coding methods (FFmpeg and HEVC Test Model, QP was

set to 32), our method required a substantially longer encoding time, even with the accelerated scalable-coding scheme. These problems should be considered in the future work.

4. Conclusion

We proposed an efficient coding scheme for a dense light field, the underlying principle of which is completely different from those of the modern image/video coding standards. Our idea is to encode a light field with several binary images and the corresponding weight values. The binary images and weight values were optimized iteratively so as to best approximate the original light field. We also mentioned

Fig. 10 Comparison of R-D curves (color) between our method (YUV method) and HEVC.

Fig. 11 Comparison of decoding time.

(a) Non-scalable coding

(b) Scalable coding

Fig. 12 Rate-distortion performance of proposed method for different resolutions. We omitted gzip compression to see the bare performance.

how this coding scheme can be made scalable and how it is applied to RGB color channels.

We demonstrated that our method has several promising properties through experiments. First, it can achieve reasonable coding efficiency (rate-distortion performance) compared with those of highly-sophisticated standard video codecs. Moreover, its decoding process is extremely simple and can be executed much faster than the standard video codecs. Furthermore, the computational complexity of the encoding process can be substantially reduced by the scalable coding scheme.

Our method is still in a preliminary form and experimental validation is limited to relatively small images due to the heavy computations required for the encoding process. For future work, we will refine our scheme and extend it to larger images, such as those taken with a Lytro Illum camera. We also plan to investigate how we handle RGB color datasets more effectively and what configurations are the best for YUV channels.

References

- [1] S. Wanner and B. Goldluecke, "Variational light field analysis for disparity estimation and super-resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.36, no.3, pp.606–619, March 2014.
- [2] T. Suzuki, K. Takahashi, and T. Fujii, "Sheared EPI analysis for disparity estimation from light fields," *IEICE Trans. Inf. & Syst.*, vol.E100-D, no.9, pp.1984–1993, Sept. 2017.
- [3] M. Levoy and P. Hanrahan, "Light field rendering," *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, pp.31–42, New York, NY, USA, ACM, 1996.
- [4] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand, "Light field reconstruction using sparsity in the continuous Fourier domain," *ACM Trans. Graph.*, vol.34, no.1, pp.12:1–12:13, Dec. 2014.
- [5] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deep stereo: Learning to predict new views from the world's imagery," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.5515–5524, June 2016.
- [6] G. Wetzstein, D. Lanman, M. Hirsch, and R. Raskar, "Tensor displays: Compressive light field synthesis using multilayer displays with directional backlighting," *ACM Trans. Graph.*, vol.31, no.4, pp.80:1–80:11, July 2012.
- [7] T. Saito, Y. Kobayashi, K. Takahashi, and T. Fujii, "Displaying real-world light fields with stacked multiplicative layers: Requirement and data conversion for input multiview images," *Journal of Display Technology*, vol.12, no.11, pp.1290–1300, Nov. 2016.
- [8] S. Lee, C. Jang, S. Moon, J. Cho, and B. Lee, "Additive light field displays: Realization of augmented reality with holographic optical elements," *ACM Trans. Graph.*, vol.35, no.4, pp.60:1–60:13, July 2016.
- [9] E.H. Adelson and J.Y.A. Wang, "Single lens stereo with a plenoptic camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.14, no.2, pp.99–106, Feb. 1992.
- [10] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report (CSTR)*, vol.2, no.11, pp.1–11, 2005.
- [11] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin, "Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing," *ACM Trans. Graph.*, vol.26, no.3, Article No.69, July 2007.
- [12] H. Nagahara, C. Zhou, T. Watanabe, H. Ishiguro, and S.K. Nayar, "Programmable aperture camera using LCoS," *Proc. 11th European*

- Conference on Computer Vision: Part VI, Lecture Notes in Computer Science, vol.6316, pp.337–350, Springer, Berlin, Heidelberg, 2010.
- [13] D.G. Dansereau, O. Pizarro, and S.B. Williams, “Decoding, calibration and rectification for lenselet-based plenoptic cameras,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1027–1034, June 2013.
 - [14] Y. Yoon, H.-G. Jeon, D. Yoo, J.-Y. Lee, and I.S. Kweon, “Learning a deep convolutional network for light-field image super-resolution,” *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp.57–65, Dec. 2015.
 - [15] N.K. Kalantari, T.-C. Wang, and R. Ramamoorthi, “Learning-based view synthesis for light field cameras,” *ACM Trans. Graph.*, vol.35, no.6, pp.193:1–193:10, Nov. 2016.
 - [16] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu, “Light field reconstruction using deep convolutional network on EPI,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1638–1646, 2017.
 - [17] G.J. Sullivan, J.M. Boyce, Y. Chen, J.-R. Ohm, C.A. Segall, and A. Vetro, “Standardized extensions of high efficiency video coding (HEVC),” *IEEE J. Sel. Topics Signal Process.*, vol.7, no.6, pp.1001–1016, Dec. 2013.
 - [18] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, “Overview of the multiview and 3D extensions of high efficiency video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol.26, no.1, pp.35–49, Jan. 2016.
 - [19] Y. Li, M. Sjöström, R. Olsson, and U. Jennehag, “Efficient intra prediction scheme for light field image compression,” *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp.539–543, May 2014.
 - [20] A. Vieira, H. Duarte, C. Perra, L. Tavora, and P. Assuncao, “Data formats for high efficiency coding of Lytro-Illum light fields,” *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp.494–497, Nov. 2015.
 - [21] D. Liu, L. Wang, L. Li, Z. Xiong, F. Wu, and W. Zeng, “Pseudo-sequence-based light field image compression,” *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp.1–4, July 2016.
 - [22] F. Hawary, C. Guillemot, D. Thoreau, and G. Boisson, “Scalable light field compression scheme using sparse reconstruction and restoration,” *2017 IEEE International Conference on Image Processing (ICIP)*, pp.3250–3254, Sept. 2017.
 - [23] X. Jiang, M.L. Pendu, R.A. Farrugia, and C. Guillemot, “Light field compression with homography-based low-rank approximation,” *IEEE J. Sel. Topics Signal Process.*, vol.11, no.7, pp.1132–1145, Oct. 2017.
 - [24] K. Komatsu, K. Takahashi, and T. Fujii, “Scalable light field coding using weighted binary images,” *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp.903–907, Oct. 2018.
 - [25] G. Koutaki, “Binary continuous image decomposition for multi-view display,” *ACM Trans. Graph.*, vol.35, no.4, pp.69:1–69:12, July 2016.
 - [26] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol.17, no.9, pp.1103–1120, Sept. 2007.
 - [27] Computer Graphics Laboratory, Stanford University, “The (new) Stanford light field archive,” <http://lightfield.stanford.edu/lfs.html>, 2008.
 - [28] K. Takahashi, “Light field compression project,” <http://www.fujii.nuee.nagoya-u.ac.jp/~takahasi/Research/LFCompression/>, 2018.
 - [29] Fraunhofer Heinrich Hertz Institute, “High efficiency video coding (HEVC),” <https://hevc.hhi.fraunhofer.de>, 2018.
 - [30] F. Bossen, “Common test conditions and software reference configurations,” *MPEG JCT-VC*, 2013.

Koji Komatsu received B.E. and M.E. degrees in Information and Communication Engineering from Nagoya University, Nagoya, Japan, in 2017 and 2019. When he was a student, his research topic was light field compression.

Kohei Isechi received his B.E. degree from National Institution for Academic Degrees and Quality Enhancement of Higher Education, Japan, in 2015, and his M.E. degree in electrical engineering and computer science from Nagoya University, Japan, in 2017. He is working on 3D video compression. He is currently a Ph.D. student in information and communication engineering at Nagoya University.

Keita Takahashi received B.E., M.S., and Ph.D. degrees in information and communication engineering from the University of Tokyo, Tokyo, Japan, in 2001, 2003, and 2006. He was a project assistant professor at the University of Tokyo from 2006 to 2011 and was an assistant professor at the University of Electro-Communications from 2011 to 2013. He is currently an associate professor at the Graduate School of Engineering, Nagoya University, Nagoya, Japan. His research interests include

computational photography, image-based rendering, and 3-D displays.

Toshiaki Fujii received B.E., M.E., and Dr.E. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1990, 1992, and 1995. In 1995, he joined the Graduate School of Engineering, Nagoya University, where he is currently a professor. From 2008 to 2010, he was with the Graduate School of Science and Engineering, Tokyo Institute of Technology. His current research interests include multidimensional signal processing, multi-camera systems, multi-view video coding

and transmission, free-viewpoint television, and their applications.