Lossless Image Coding Exploiting Local and Non-local Information via Probability Model Optimization

Kyohei Unno*†, Yusuke Kameda†, Ichiro Matsuda†, Susumu Itoh†, and Sei Naito*
*KDDI Research, Inc.
2-1-15 Ohara, Fujimino-shi, Saitama 356-8502, JAPAN
†Faculty of Science and Technology, Tokyo University of Science
2641 Yamazaki, Noda-shi, Chiba 278-8510, JAPAN
Email: ky-unno@kddi-research.jp

***Abstract*—We previously proposed a lossless image coding method based on examples search and probability model optimization. In this paper, we improve coding efficiency of the method by introducing an adaptive prediction technique. Specifically, multiple affine predictors are trained pel-by-pel by using causal neighbor pels, and the predicted values obtained by those predictors are used for generating the probability model. Therefore, both non-local information by the examples search and local information by the adaptive prediction are used together in the probability modeling. Furthermore, an optimization method for the number of examples is also proposed in this paper. Experimental results show that the proposed method achieves better coding rates than the state-of-the-art lossless coding schemes.**

***Index Terms*—Lossless coding, Example search, Affine prediction, Probability modeling, Quasi-Newton method**

I. Introduction

Most image coding schemes consist of two stages. One is a decorrelation stage, and the other is an entropy coding stage. In this framework, decorrelate signals generated by the first stage, i.e. prediction residuals or transform coefficients, are expected to follow a symmetric single-peaked distribution centered around zero. In predictive coding, the least-square method [1], or weighted least-square method [2], [3] is often used to design adaptive predictors in the first stage for reducing entropy. On the other hand, our previous approach iteratively optimizes multiple predictors to directly minimize the coding rate of the resulting prediction residuals [4]. Moreover, Gaussian process regression in [5] and deep neural networks in [6] are used respectively instead of linear prediction to generate the predicted values. In all those cases, probability distributions used in an entropy coding stage are assumed to be single-peaked, and modeled in a decorrelated signal domain.

Recently, we developed a new coding scheme that estimates probability distributions of luma sample values without the decorrelation stage. In [7], probability distributions of luma sample values are approximated by a Gaussian mixture model and coded by a multilevel arithmetic coding technique. Mean values of the respective Gaussian distributions are estimated by examples that are collected from a causal neighborhood by template matching. The shape of the probability distributions is controlled by parametric functions associated with the reliability of the examples. Some parameters used in the functions are numerically optimized by the quasi-Newton method to minimize the coding rate, and then transmitted to the decoder side as side-information.

A similar approach was seen in [8] from aspects of utilizing multi-peaked probability distribution. However, peak positions of probability distributions are estimated by only using multiple linear predictors. In other words, the method uses only local information for the probability modeling.

In this paper, we propose utilizing predicted values from the local area in addition to the examples collected from the non-local area. Concretely, multiple affine predictors are designed for each pel, and the predicted values are used together with the examples for estimating the probability distribution.

II. Examples Search and Optimization of Probability Models

In this section, the lossless image coding method proposed in [7] is explained as a base method of the proposal.

In the base method, M types of examples, which are expected to provide non-local information of image signals, are collected by template matching from the causal neighborhood. Figure 1 illustrates this examples search process. In this figure, $\boldsymbol{p}_k$ indicates a target pel that is processed k-th in raster scan order. The search window of the template matching is limited within a distance of $S = 80$ pels. A template consists of twelve pels $\{\boldsymbol{p}_k + \boldsymbol{r}_i \,|\, i = 1, 2, \ldots, 12\}$ located within $\|\boldsymbol{r}_i\|_1 \leq 3$. Here, $\boldsymbol{r}_i$ is the relative position from $\boldsymbol{p}_k$. The cost function of template matching is defined as (1).

$$J_k(\boldsymbol{q}) = \left[\sum_{i=1}^{12} w_i \cdot \big(f(\boldsymbol{q}+\boldsymbol{r}_i) - \mu(\boldsymbol{q}) - f(\boldsymbol{p}_k+\boldsymbol{r}_i) + \mu(\boldsymbol{p}_k)\big)^2\right]^{\frac{1}{2}} + \lambda_d \cdot \|\boldsymbol{q} - \boldsymbol{p}_k\|_1, \quad (1)$$

$$\mu(\boldsymbol{q}) = \sum_{i=1}^{12} w_i \cdot f(\boldsymbol{q} + \boldsymbol{r}_i), \quad (2)$$

978-9-0827-9703-9/19/$31.00

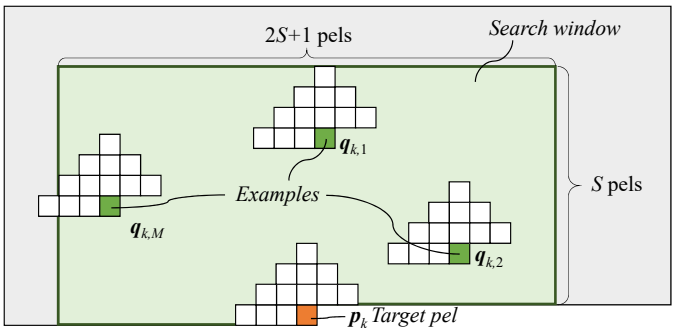

Fig. 1. Examples search.

$$w_i = \frac{\exp(-\frac{1}{2}\|\boldsymbol{r}_i\|_1^2/\sigma_t^2)}{\sum_{l=1}^{12}\exp(-\frac{1}{2}\|\boldsymbol{r}_l\|_1^2/\sigma_t^2)}. \tag{3}$$

The first term on the right-hand side of (1) means similarity of textures measured by weighted root mean squared differences. $f(\boldsymbol{q})$ is the luma sample value of pel $\boldsymbol{q}$, $\mu(\boldsymbol{q})$ is a weighted local mean of the luma sample values and w_i is a weighting factor defined by the Gaussian function with $\sigma_t = 1.25$. The second term imposes a penalty for the likelihood of selecting examples near the target pel. A weighting factor of $\lambda_d = 0.03$ is determined based on the experiment reported in [7]. Values of the cost function (1) are calculated for all pels in the search window. Then, M pels $\{\boldsymbol{q}_{k,1}, \boldsymbol{q}_{k,2}, \ldots, \boldsymbol{q}_{k,M}\}$ that have smaller cost function values are collected as examples.

After the examples search, probability distributions are estimated based on the examples. Those are modeled by a linear combination of Gaussian functions.

$$\Pr(f \,|\, \boldsymbol{E}_k, u_k) \propto P(f \,|\, \boldsymbol{E}_k, u_k) = \sum_{m=1}^{M} g_{k,m}(f) + \epsilon, \tag{4}$$

$$g_{k,m}(f) = h_{k,m} \cdot w_{k,m} \cdot \exp\left(-w_{k,m}^2 \cdot (f - f_{k,m})^2\right), \tag{5}$$

where $\boldsymbol{E}_k$ means a set of examples $\{\boldsymbol{q}_{k,1}, \boldsymbol{q}_{k,2}, \ldots, \boldsymbol{q}_{k,M}\}$, $f_{k,m}$ is a luma sample value given by the m-th example $\boldsymbol{q}_{k,m}$ after compensating its local mean value.

$$\begin{aligned} f_{k,m} \;=\; & f(\boldsymbol{q}_{k,m}) - \mu(\boldsymbol{q}_{k,m}) + \mu(\boldsymbol{p}_k), \\ & (m = 1, 2, \ldots, M). \end{aligned} \tag{6}$$

u_k is a feature value defined by neighboring pels' number of coded bits, and $\epsilon = 2^{-20}$ is a positive constant value added to avoid the probability being zero. Moreover, $h_{k,m}$ and $w_{k,m}$ are parameters that control the height and width of each Gaussian as shown in Fig. 2. We consider these parameters should reflect reliability of the examples and define them as parametric functions of the matching cost $d_{k,m} = J_k(\boldsymbol{q}_{k,m})$ as well as u_k.

$$h_{k,m} = \exp(-a_1 \cdot d_{k,m}), \tag{7}$$

$$w_{k,m} = a_0 \cdot \exp(-a_2 \cdot d_{k,m}) \cdot \exp(-a_3 \cdot u_k). \tag{8}$$

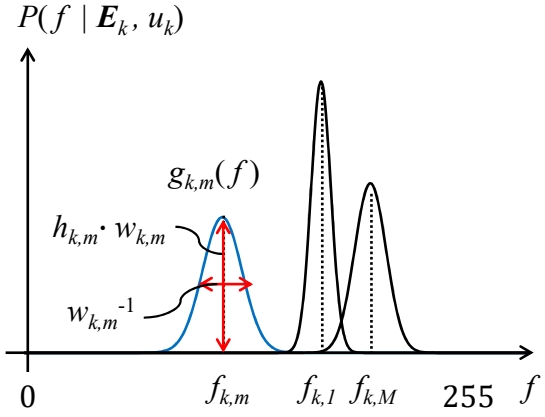

Fig. 2. Probability distribution model.

According to this model, the probabilities of possible luma sample values ($f = 0, 1, \ldots, 255$ for 8 bits image) of the target pel can be calculated by normalization of (4).

$$\Pr(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, u_k) = \frac{P(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, u_k)}{\sum_{f=0}^{255} P(f \,|\, \boldsymbol{E}_k, u_k)}. \tag{9}$$

Thus, the number of coded bits of $\boldsymbol{p}_k$ can be estimated by:

$$\begin{aligned} \mathcal{L}(\boldsymbol{p}_k) \;&=\; -\log_2 \Pr(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, u_k) \\ &=\; \frac{1}{\ln 2}\Bigg[\ln\Bigg(\sum_{f=0}^{255} P(f \,|\, \boldsymbol{E}_k, u_k)\Bigg) \\ &\qquad - \ln P(f(\boldsymbol{p}_k) | \boldsymbol{E}_k, u_k)\Bigg]. \end{aligned} \tag{10}$$

This estimation can be seen as a consistency of the probability model. Therefore, the feature value u_k is defined as (11) to feedback the accuracy of the probability model.

$$u_k = \sum_{i=1}^{12} w_i \cdot \mathcal{L}(\boldsymbol{p}_k + \boldsymbol{r}_i). \tag{11}$$

In the base method, the shape of a probability distribution function is determined by four model parameters $a_0, \ldots, a_3$. They are numerically optimized to minimize the number of coded bits in each region $\boldsymbol{\Omega}$ of 64×64 pels by the quasi-Newton method, and then sent to the decoder-side with a fixed length code of 8 bits as side-information [7].

In this paper, the number of examples M, which was fixed to 64 in the base method, is also optimized in each region of the same size. The optimization scheme is described in **IV-B** and **V**.

III. Introduction of Adaptive Prediction

This paper proposes the use of pel-wise adaptive prediction for capturing local information. Namely, locally obtained predicted values $f_{k,n}$ $(n = 1, \ldots, N)$ are used in addition to $f_{k,m}$ $(m = 1, \ldots, M)$ for the probability modeling. These predicted values are calculated by N different affine predictors. Prediction coefficients $b_{n,l}$ $(l = 0, \ldots, L_n)$ of each predictor are trained pel-by-pel to minimize (12) in a region $\boldsymbol{T}$ shown in

Fig. 3. In this paper, we set the size of the training region $\boldsymbol{T}$ to $R = 10$ based on the results of our preliminary experiments.

$$e_n = \sum_{\boldsymbol{p}_i \in \boldsymbol{T}} \frac{1}{\sigma_{n,i}^2} \left(f(\boldsymbol{p}_i) - \left(\sum_{l=1}^{L_n} b_{n,l} \cdot f(\boldsymbol{p}_i + \boldsymbol{r}'_{n,l}) + b_{n,0} \right) \right)^2. \quad (12)$$

Here, L_n is the number of reference pels for the n-th predictor and $b_{n,0}$ is a translation term. $\boldsymbol{r}'_{n,l}$ $(l = 1, 2, \ldots, L_n)$ means vectors that define the reference pel arrangement of the respective predictors. $\sigma_{n,i}^2$ shows the mean squared differences between reference pels of $\boldsymbol{p}_k$ and $\boldsymbol{p}_i$.

$$\sigma_{n,i}^2 = \frac{1}{L_n} \sum_{l=1}^{L_n} \left(f(\boldsymbol{p}_k + \boldsymbol{r}'_{n,l}) - f(\boldsymbol{p}_i + \boldsymbol{r}'_{n,l}) \right)^2. \quad (13)$$

In practice, the minimum value of $\sigma_{n,i}^2$ is clipped by $1/64$ to avoid zero division in (12).

While the predicted values are used as $f_{k,n}$ in (5), regression errors in the training region are also used as $d_{k,n}$ in (7) and (8). They are calculated by the trained affine predictors.

$$f_{k,n} = \sum_{l=1}^{L_n} b_{n,l} \cdot f(\boldsymbol{p}_k + \boldsymbol{r}'_{n,l}) + b_{n,0}, \quad (14)$$

$$d_{k,n} = \left(\frac{e'_n}{\sum_{\boldsymbol{p}_i \in \boldsymbol{T}} 1/\sigma_{n,i}^2} \right)^{\frac{1}{2}}, \quad (15)$$

where, e'_n means a minimized prediction error in (12). On the other hand, the feature value u_k can be different values for each predictor depending on its reference pels arrangement:

$$u_{k,n} = \sum_{i=1}^{L_n} w_{n,i} \cdot \mathcal{L}(\boldsymbol{p}_k + \boldsymbol{r}'_{n,i}), \quad (16)$$

$$w_{n,i} = \frac{\exp(-\frac{1}{2}\|\boldsymbol{r}_{n,i}\|_1^2/\sigma_t^2)}{\sum_{l=1}^{L_n} \exp(-\frac{1}{2}\|\boldsymbol{r}_{n,l}\|_1^2/\sigma_t^2)}. \quad (17)$$

Therefore, we defined $\boldsymbol{u}_k$ as a set of feature parameters $\{u_k, u_{k,1}, \ldots, u_{k,N}\}$.

From the above, $P(f \mid \boldsymbol{E}_k, u_k)$ in (4) is changed as follows.

$$P(f \mid \boldsymbol{E}_k, \boldsymbol{u}_k) = \sum_{m=1}^{M} g_{k,m}(f) + \sum_{n=1}^{N} g_{k,n}(f) + \epsilon. \quad (18)$$

IV. Setting of Parameters M and N

As described above, probability distributions of luma sample values are modeled by a linear combination of $M + N$ types of Gaussian functions. In this paper, M is optimized for each region $\boldsymbol{\Omega}$ of 64×64 pels as an encoding parameter just like $a_0, \ldots, a_3$ in [7]. On the other hand, N is fixed to 25. The details are as follows.

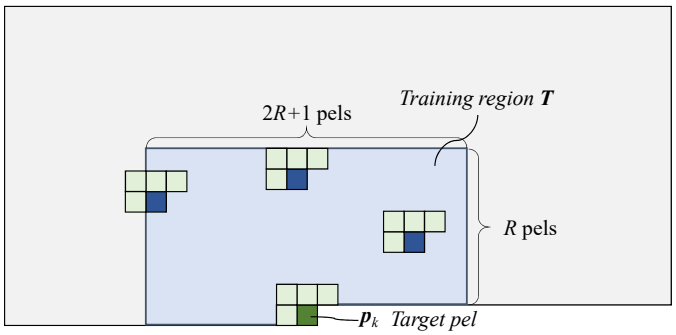

Fig. 3. Training region for adaptive predictors.

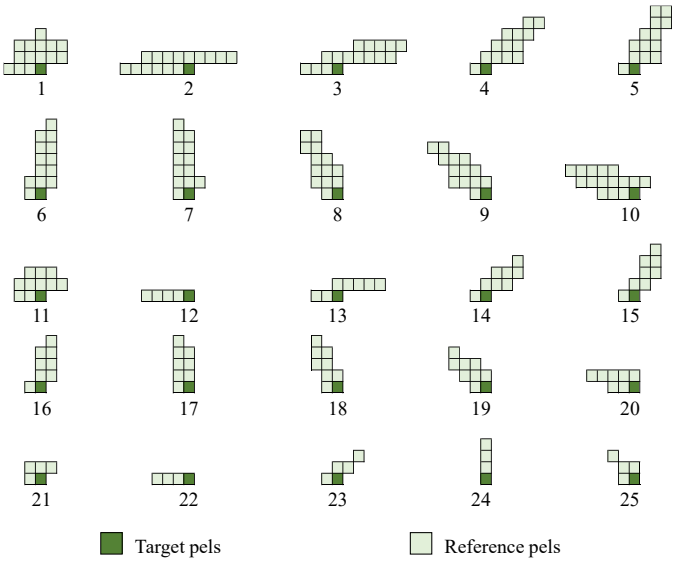

Fig. 4. Reference pels for respective predictors.

A. The number of predictors N

In this paper, we exploit $N = 25$ types of affine predictors for each pel. The value of N and arrangement of reference pels are determined in an experimental approach. An ellipse given by (19) where $\boldsymbol{p}_k$ corresponds to its origin is used for this purpose. Concretely, integer positions (x, y) in the ellipse defined by parameters β, γ and θ, are picked up and used as $\{\boldsymbol{r}'_{n,l}\}$ if they are within the causal neighborhood region. In this way, N types of arrangements are determined by N combinations of β, γ and θ to fit various local textures. The resulting arrangements are shown in Fig. 4.

$$O(x, y) = \beta \cdot \{(\gamma \cdot c_1)^2 + (c_2)^2\} \leq 1 \quad (19)$$
$$\begin{cases} c_1 = x \cdot \sin\theta + y \cdot \cos\theta, \\ c_2 = x \cdot \cos\theta - y \cdot \sin\theta. \end{cases}$$

Optimal arrangements of reference pels as well as the setting of N are different for each image and each region depending on textures. We tried to change only N in an adaptive way, but coding rates were not improved. This suggests us that not only N but also combinations of β, γ and θ must be optimized together. It is not a straightforward process and will be studied in future.

B. The number of examples M

M is optimized in each region $\mathbf{\Omega}$ within a range of 0 to M_{max}. If M is set to 0, no examples are used for modeling the probability distribution in that region (i.e. only $N = 25$ types of predictors are used). The optimization can be achieved in the following manner.

At first, M is initialized by a value of M_{max}. In this paper, we set M_{max} to 31. In the case of fixed M with $N = 25$, our preliminary experiment shows the best setting of M is 7. Therefore, we think $M_{max} = 31$ is a sufficient number. After updating the model parameters $a_0, \ldots, a_3$ using the quasi-Newton method, the probability modeling is performed again by temporarily restricting the number of examples to M' $(M' = 0, \ldots, M)$.

$$P_{M'}(f \,|\, \boldsymbol{E}_k, \boldsymbol{u}_k) = \tag{20}$$
$$\begin{cases} \displaystyle\sum_{n=1}^{N} g_{k,n}(f) + \epsilon & (\text{if } M' = 0), \\ \displaystyle\sum_{m=1}^{M'} g_{k,m}(f) + \sum_{n=1}^{N} g_{k,n}(f) + \epsilon & (\text{otherwise}), \end{cases}$$

$$\Pr_{M'}(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, \boldsymbol{u}_k) = \frac{P_{M'}(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, \boldsymbol{u}_k)}{\sum_{f=0}^{255} P_{M'}(f \,|\, \boldsymbol{E}_k, \boldsymbol{u}_k)}. \tag{21}$$

Under this restriction, the total number of coded bits in the region $\mathbf{\Omega}$ can be estimated by:

$$\mathcal{L}(\mathbf{\Omega}, M') = -\sum_{\boldsymbol{p}_k \in \mathbf{\Omega}} \log_2 \Pr_{M'}(f(\boldsymbol{p}_k) \,|\, \boldsymbol{E}_k, \boldsymbol{u}_k). \tag{22}$$

The calculation of (22) for successive values of M' can be effectively performed by storing and updating both the numerator and denominator of (21) while adding the Gaussian function $g_{k,m}(\boldsymbol{p}_k)$ one by one in (20). Thus we can optimize M to minimize the total bits.

$$M = \underset{M'}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{\Omega}, M'). \tag{23}$$

The final value of M is sent to the decoder side with a fixed length code of 5 bits as side-information.

V. Processing Procedure

The processing procedure at the encoder side is summarized as follows.

a) The initial value of M is set to $M_{max} = 31$, and $a_0, \ldots, a_3$ are set to the same initial values as [7].
b) M types of examples are collected as described in **II**, then the values of $f_{k,m}$ and d_k $(m = 1, \ldots, M)$ are stored.
c) N types of affine predictors are trained as described in **III**, then values of $f_{k,n}$, $d_{k,n}$ and $u_{k,n}$ $(n = 1, \ldots, N)$ are stored.
d) The above b) and c) are processed for each pel in raster scan order.
e) When the above processes have been completed for 64 lines, $a_0, \ldots a_3$ and M are optimized for each region $\mathbf{\Omega}$ obtained by dividing those lines at even intervals.
i) Probability distribution models (18) are calculated.
ii) $a_0, \ldots, a_3$ are updated by the quasi-Newton method [9].
iii) The above i) and ii) are iterated until conversion or maximum iteration steps (10 times in this paper).
iv) Elements of $\boldsymbol{u}_k$ are updated for every pel.
v) M is updated as described in **IV-B**.
vi) The above i),. . . ,v) are iterated at most 5 times, then model parameters $a_0, \ldots, a_3$ are linear quantized.
vii) M and $a_0, \ldots, a_3$ are encoded as side-information.
f) The probability (18) is calculated for each $\boldsymbol{p}_k$.
g) The actual value of $f(\boldsymbol{p}_k)$ is encoded by arithmetic coding using the obtained probability model. A range coder [10] is employed in our implementation.
h) The above procedures are exploited for all pels in the given image.

At the decoder side, $a_0, \ldots a_3$ and M are decoded as a first step. Then examples are collected and predicted values are generated as in b) and c) at each pel, and the luma sample value is decoded by estimating probabilities as in f).

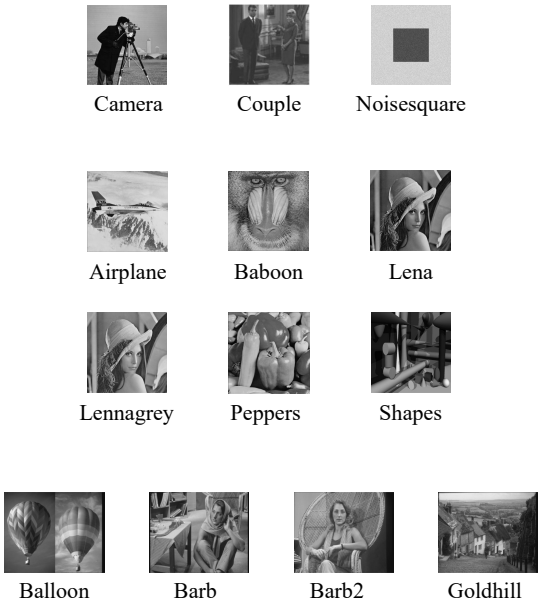

Fig. 5. Test images.

VI. Experimental Results

To verify the efficiency of the proposed method, we conducted coding simulations for gray scale still images shown in Fig. 5. The coding rates are reported in Table I. Here, "Base" means the base method [7] with the fixed number of examples $(M = 64)$. Other compared methods are as follows: MRP (version 0.5) [4], Vanilc WLS D (version 1.0) [11], TMW (version 0.51) [8], Glicbawls [3], WebP lossless (version 0.6.0) [12], BMF [13], JPEG-LS [14], and JPEG 2000 [15].

TABLE I
COMPARISON OF CODING RATES (BITS/PEL).

Image	Size	Proposed	Base	MRP	Vanilc	TMW	BMF	Glicbawls	WebP	JPEG-LS	JPEG 2000
Camera	256×256	**3.833**	3.960	3.949	3.995	4.098	3.952	4.208	4.274	4.314	4.535
Couple		**3.281**	3.415	3.388	3.459	3.446	3.375	3.543	3.703	3.699	3.915
Noisesquare		5.296	5.298	5.270	**5.159**	5.542	5.238	5.415	5.203	5.683	5.634
Airplane	512×512	3.546	3.632	3.591	3.575	3.601	**3.535**	3.668	3.894	3.817	4.013
Baboon		5.698	5.727	**5.663**	5.678	5.738	5.677	5.666	5.891	6.037	6.107
Lena		**4.237**	4.330	4.280	4.246	4.300	4.252	4.295	4.514	4.607	4.684
Lennagrey		**3.845**	3.944	3.889	3.856	3.908	3.863	3.901	4.145	4.238	4.303
Peppers		**4.176**	4.267	4.199	4.187	4.251	4.177	4.246	4.495	4.513	4.629
Shapes		**0.497**	0.715	0.685	1.302	0.740	0.702	2.291	1.023	1.214	1.926
Balloon	720×576	2.584	2.673	2.579	2.626	2.649	**2.560**	2.640	2.925	2.904	3.031
Barb		**3.733**	3.997	3.815	3.815	4.084	3.804	3.916	4.547	4.691	4.600
Barb2		**4.146**	4.287	4.216	4.231	4.378	4.163	4.318	4.668	4.686	4.789
Goldhill		4.191	4.276	4.207	4.229	4.266	**4.179**	4.276	4.464	4.477	4.603
Average		**3.774**	3.886	3.826	3.874	3.923	3.806	4.030	4.134	4.222	4.367

In Table I, the best coding rates are shown as **bold** numbers for each image. The coding rates of the proposed method are improved for all test images by using local information (from adaptive prediction) in addition to non-local information (by examples search) in comparison with the "Base". Additionally, the proposed method achieves the lowest coding rates for most of the test images and for "Average". The proposed method archives good performances for images with clear edges such as "Camera" and "Shapes". However, some conventional methods archive lower coding rates than the proposed method for images with low spatial correlation such as "Noisesquare" and "Baboon". To improve the performances for those images, one solution may be to change the probability distribution model from Gaussian to another version. For example, the t-distribution could be an option since it is often used for probability modeling as in [8] and [11].

As reference, we measured processing time of the proposed method for 512×512 size images on a computer with Intel Xeon@2.60Ghz CPU and 64GB memory. Average encoding time was about 102 minutes, and the most part was spent for the optimization process in **V-e)** (84 min.). Average decoding time was about 18 minutes, because the optimization is not necessary at the decoder. Processing time reduction as well as extension to color images will be our future tasks.

VII. CONCLUSION

We have improved our previous lossless image coding method (referred to as the base method) by introducing an adaptive prediction technique. The base method estimates probability distributions of luma sample values based on examples collected from the causal neighborhood. Moreover, some parameters that control shapes of the probability model are optimized to reduce the number of coded bits. The proposed method uses multiple affine predictors together with the examples for the probability model optimization. Experimental results show that the proposed method achieves the lowest coding rates for most of the tested images in comparison with the state-of-the-art lossless still image coding methods.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 17K00247.

REFERENCES

[1] L.-J. Kau and Y.-P. Lin, "Adaptive Lossless Image Coding Using Least Squares Optimization With Edge-Look-Ahead," *IEEE Trans. on Circuits and Systems II: Express Briefs*, Vol.52, No.11, pp.751–755, Nov. 2005.

[2] H. Ye, G. Deng and J. C. Devlin, "A Weighted Least Squares Method for Adaptive Prediction in Lossless Image Compression," *Proc. 2003 Picture Coding Symposium (PCS 2003)*, pp.489–493, Apr. 2003.

[3] B. Meyer and P. Tischer, "Glicbawls — Grey Level Image Compression by Adaptive Weighted Least Squares," *Proc. 2001 Data Compression Conf. (DCC 2001)*, p.503, Mar. 2001.

[4] I. Matsuda, N. Ozaki, Y. Umezu and S. Itoh, "Lossless Coding Using Variable Block-Size Adaptive Prediction Optimized for Each Image," *Proc. 13th European Signal Processing Conf. (EUSIPCO 2005)*, Sep. 2005.

[5] W. Dai and H. Xiong, "Gaussian Process Regression Based Prediction for Lossless Image Coding," *Proc. 2014 Data Compression Conf. (DCC 2014)*, pp.93–102, Mar. 2014.

[6] I. Schiopu, Y. Liu and A. Munteanu, "CNN-based Prediction for Lossless Image Coding of Photographic Images," *Proc. 2018 Picture Coding Symposium (PCS 2018)*, pp.16–20, June 2018.

[7] I. Matsuda, T. Ishikawa, Y. Kameda and S. Itoh, "A Lossless Image Coding Method Based on Probability Model Optimization," *Proc. 26th European Signal Processing Conf. (EUSIPCO 2018)*, pp.156–160, Sep. 2018.

[8] B. Meyer and P. Tischer, "TMW — A New Method for Lossless Image Compression," *Proc. 1997 Picture Coding Symposium (PCS '97)*, pp.533–538, Sep. 1997.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B.P. Flannery, "Numerical Recipes: The Art of Scientific Computing, Third Edition," *Cambridge University Press*, 2007.

[10] M. Schindler, "A Fast Renormalization for Arithmetic Coding," *Proc. Data Compression Conf. (DCC '98)*, p.572, 1998.

[11] A. Weinlich, P. Amon, A. Hutter and A. Kaup, "Probability Distribution Estimation for Autoregressive Pixel-Predictive Image Coding," *IEEE Trans. on Image Processing*, Vol.25, No.3, pp.1382–1395. Mar. 2016.

[12] Google Developers, "A new image format for the web," http://developers.google.com/speed/webp/.

[13] http://compression.ru/ds/.

[14] ISO/IEC, 14495-1:1999, "Information Technology — Lossless and Near-Lossless Compression of Continuous-Tone Still Images: Baseline," Dec. 1999.

[15] ITU-T Rec. T.800 — ISO/IEC 15444-1, "Information Technology — JPEG 2000 Image Coding System – Part 1: Core Coding System," 2001.