Edge-Aware Extended Star-Tetrix Transforms for CFA-Sampled Raw Camera Image Compression

Taizo Suzuki, *Senior Member, IEEE*, and Liping Huang, *Student Member, IEEE*

***Abstract*—Codecs using spectral-spatial transforms efficiently compress raw camera images captured with a color filter array (CFA-sampled raw images) by changing their RGB color space into a decorrelated color space. This study describes two types of spectral-spatial transform, called extended Star-Tetrix transforms (XSTTs), and their edge-aware versions, called edge-aware XSTTs (EXSTTs), with no extra bits (side information) and little extra complexity. They are obtained by (i) extending the Star-Tetrix transform (STT), which is one of the latest spectral-spatial transforms, to a new version of our previously proposed wavelet-based spectral-spatial transform and a simpler version, (ii) considering that each 2-D predict step of the wavelet transform is a combination of two 1-D diagonal or horizontal-vertical transforms, and (iii) weighting the transforms along the edge directions in the images. Compared with XSTTs, the EXSTTs can decorrelate CFA-sampled raw images well: they reduce the difference in energy between the two green components by about 3.38–30.08 % for high-quality camera images and 8.97–14.47 % for mobile phone images. The experiments on JPEG 2000-based lossless and lossy compression of CFA-sampled raw images show better performance than conventional methods. For high-quality camera images, the XSTTs/EXSTTs produce results equal to or better than the conventional methods: especially for images with many edges, the type-I EXSTT improves them by about 0.03–0.19 bpp in average lossless bitrate and the XSTTs improve them by about 0.16–0.96 dB in average Bjøntegaard delta peak signal-to-noise ratio. For mobile phone images, our previous work perform the best, whereas the XSTTs/EXSTTs show similar trends to the case of high-quality camera images.**

***Index Terms*—Color filter array, edge-aware, raw camera image compression, spectral-spatial transforms, wavelet transforms.**

I. Introduction

RAW camera images are mainly created by placing a color filter array (CFA) between the light sensors and the camera lens. To economize on hardware, most cameras capture a color image with a single sensor instead of using three RGB sensors. In other words, each pixel of such a sensor collects a single color component, either red, green, or blue, not all three, and the obtained raw data is called a CFA-sampled raw image. The Bayer CFA is the most popular type (Fig. 1). Using a CFA-sampled raw image as is, an image processor performs most of the preprocessing, including black level correction, white balance, demosaicing [1–4], and gamma correction. In particular, the image quality largely depends on the performance of the demosaicing process that produces the full-color RGB image we see. Moreover, performing compression after demosaicing (the demosaicing-first approach) causes redundancy wherein the data volume of a full-color RGB image is three times that of the original CFA-sampled raw image, whereas performing compression before demosaicing (the compression-first approach) can avoid this redundancy. This compression-first approach also allows us to process images more freely than the demosaicing-first one which almost automatically performs various image processings before compression. In light of these facts, although many traditional image-compression methods take the demosaicing-first approach, as do standards such as JPEG [5] and JPEG 2000 [6], JPEG XS 2nd Edition [7] takes both approaches. We believe that the compression-first approach will get more attention in the future from not only professional photographers and designers but also typical consumers.

This work was supported by JSPS Grant-in-Aid for Scientific Research (C) under Grant 22K04084.

T. Suzuki is with the Faculty of Engineering, Information and Systems, University of Tsukuba, Ibaraki 305-8573, Japan (e-mail: taizo@cs.tsukuba.ac.jp).

L. Huang is with Graduate School of Science and Technology, University of Tsukuba, Ibaraki 305-8573, Japan (e-mail: huang@wmp.cs.tsukuba.ac.jp).

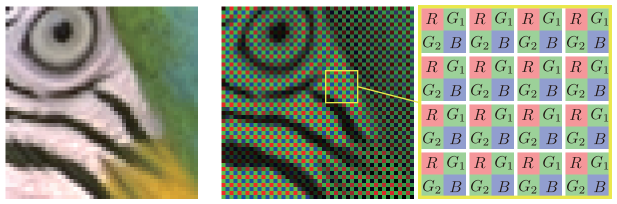

Fig. 1. Bayer pattern of a particular area of *Parrot* in the Kodak images dataset [8] (each 2×2 pixel square is a macropixel): (left) RGB full-color RGB image and (right) simulated CFA-sampled raw image with corresponding diagram.

Spectral-spatial transforms for the compression-first approach have been widely researched [9–15]. They change a CFA-sampled raw image in RGB color space into data in a decorrelated color space, such as the YDgCbCr or YDgCoCg color space composed of luma, difference green, and two chroma components. The spectral redundancy between the decorrelated components is very small. Since the human visual system is not very sensitive to distortion of high-frequency (different green) and chroma components, the strong compression of the decorrelated components will not affect the image quality much. This study focuses on our previous work on wavelet-based spectral-spatial transforms (WSSTs) [14] that are represented by cascading (discrete) wavelet transforms, such as Haar, 5/3, and 9/7 wavelet transforms, and cover the other spectral-spatial transforms [9–12]. Additionally, there are several new spectral-spatial transforms that are not described in [14]. One is a newer spectral-spatial transform presented by Lee et al. in [13] for a CFA-sampled raw image coding framework, called camera-aware multi-resolution analysis (CAMRA) [12]. The YDgCoCg2-WSSTs in [14] generalize the transform in [13]: the transform in [13] is obtained by simply applying a 5/3 wavelet transform instead of a Haar transform between the LH and HL subbands to the existing transforms in [12]. The other one is the Star-Tetrix transform (STT) of the low-latency low-complexity image coding standard, JPEG XS, developed by Richter et al. [15]. It is constructed from three 5/3 wavelet transforms. The pixels used in the predict and update steps are not limited to be within a macropixel; the surrounding pixels are also used to make the predict results more accurate, as in the case of the existing WSSTs. On the other hand, for full-color RGB images, (non-redundant) adaptive directional wavelet transforms [16–18], which adapt the filtering directions along the edge information, are efficient and popular methods that take into account image features. Note that, compared with directional transforms, direct use of wavelet transforms for any of the existing spectral-spatial transforms amounts to nothing but ignoring the image features, especially, edge information. However, the existing directional transforms do so with extra bits (side information), which adversely affect coding performance, and a significant amount of complexity.

This study develops extended STT (XSTTs) and edge-aware XSTTs (EXSTTs) with no side information and little extra complexity. They are obtained by (i) extending the STT [15] to a new version of the WSSTs [14] and a simpler version, (ii) considering that each 2-D predict step of the wavelet transforms is a combination of two 1-D diagonal or horizontal-vertical transforms, and (iii) weighting the transforms along the edge directions in the images. Compared with XSTTs, the EXSTTs can decorrelate CFA-sampled raw images well. In experiments on JPEG 2000-based lossless and lossy compression of high-quality camera images, our XSTTs/EXSTTs produce results equal to or better than the conventional methods. Especially for images with many edges, our XSTTs/EXSTTs outperform the conventional methods because of their more efficient decorrelation. Note that in lossy compression, unlike lossless compression, it is more practical to use XSTTs instead of EXSTTs because the lossy weights are re-calculated and used in the decoder. In experiments on mobile phone images, which are relatively noisy, our previous work performs the best, whereas the XSTTs/EXSTTs show similar trends to the case of high-quality camera images.

A preliminary version of this study was presented in [19], where we discussed only the edge-aware weighted 2-D diagonal predict steps for the YDgCoCg-WSSTs in [14]. In addition to that, this paper extends the STTs in [15] to XSTTs and further presents the edge-aware weighted 2-D horizontal-vertical predict steps for the XSTTs.

The remainder of the paper is organized as follows. Section II reviews the conventional methods: WSSTs and STT. Section III extends the STT to two new versions of the WSSTs, i.e., the XSTTs, and introduces two types of edge-aware weighted 2-D predict steps to be applied to the XSTTs. Section IV shows the effect of the edge-aware weighted 2-D predict steps and compares the resulting XSTTs/EXSTTs with the conventional methods in JPEG 2000 for CFA-sampled raw image compression. Section V concludes the paper.

Notation: Boldface letters represent vectors and matrices. $\mathbf{I}$ and $\mathbf{O}$ denote a 2×2 identity matrix and zero matrix, respectively. Moreover, $\cdot^\top$, $\lfloor \cdot \rfloor$, and $| \cdot |$ denote the transpose, floor function (rounding operation), and absolute value, respectively. Let z_i be a horizontal ($i = 1$) or vertical ($i = 2$) delay element and $\bar{z}_i = z_i^{-1}$. In addition, the size and dynamic range of the images in the figures have been adjusted for display.

TABLE I
COEFFICIENTS OF 5/3 AND 9/7 WAVELET TRANSFORMS.

	5/3	9/7
p_0	$-1/2$	-1.58613434205992
u_0	$1/4$	-0.05298011857295
p_1	0	0.882911075530940
u_1	0	0.443506852043967

II. Review and Definitions

A. Wavelet-Based Spectral-Spatial Transforms

Our previous work [14] presented WSSTs that cover many spectral-spatial transforms for CFA-sampled raw image compression. The WSST $\mathfrak{T}$ is represented as

$$[Y, D_g, C_1, C_2]^\top = \mathfrak{T}\, [G_1, G_2, B, R]^\top , \tag{1}$$

where R, G_1, G_2, B, Y, D_g, C_1, and C_2 mean red, green, other green, blue, luma, difference green, chroma, and other chroma components, respectively. The $\mathfrak{T}$s are classified into three types: YDgCbCr-WSSTs $\mathfrak{T}_{\mathrm{br}}$, YDgCoCg-WSSTs $\mathfrak{T}_{\mathrm{og}}$, and YDgCoCg2-WSSTs $\mathfrak{T}_{\mathrm{og2}}$, as follows:

$$\mathfrak{T}_{\mathrm{br}} = \mathbf{P}_0 \begin{bmatrix} 1 & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\mathcal{W}}_3(\bar{z}_1, z_2) \end{bmatrix} \mathbf{P}_0 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\bar{z}_1, z_2) & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}, \tag{2}$$

$$\mathfrak{T}_{\mathrm{og}} = \mathbf{P}_2 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\bar{z}_2) & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \mathbf{P}_1 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\bar{z}_1, z_2) & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\mathcal{W}}_2(\bar{z}_1, \bar{z}_2) \end{bmatrix}, \tag{3}$$

$$\mathfrak{T}_{\mathrm{og2}} = \mathbf{P}_5 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\bar{z}_1, z_2) & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \mathbf{P}_4 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(z_1) & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\mathcal{W}}_2(z_1) \end{bmatrix} \cdot \mathbf{P}_3 \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(z_2) & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\mathcal{W}}_2(z_2) \end{bmatrix} \mathbf{P}_2, \tag{4}$$

where $\boldsymbol{\mathcal{W}}_2(z_i)$, $\boldsymbol{\mathcal{W}}_2(z_1, z_2)$, and $\boldsymbol{\mathcal{W}}_3(z_1, z_2)$ are wavelet transforms:

$$\boldsymbol{\mathcal{W}}_2(z_i) = \prod_{k=N-1}^{0} \underbrace{\begin{bmatrix} 1 & U_k(z_i) \\ 0 & 1 \end{bmatrix}}_{\text{update step}} \underbrace{\begin{bmatrix} 1 & 0 \\ P_k(z_i) & 1 \end{bmatrix}}_{\text{predict step}}, \tag{5}$$

$$\boldsymbol{\mathcal{W}}_2(z_1, z_2) = \prod_{k=N-1}^{0} \underbrace{\begin{bmatrix} 1 & U_k(z_1, z_2) \\ 0 & 1 \end{bmatrix}}_{\text{update step}} \underbrace{\begin{bmatrix} 1 & 0 \\ P_k(z_1, z_2) & 1 \end{bmatrix}}_{\text{predict step}}, \tag{6}$$

$$\boldsymbol{\mathcal{W}}_3(z_1, z_2) = \prod_{k=N-1}^{0} \underbrace{\begin{bmatrix} 1 & \mathbf{O} \\ \frac{1}{2}\begin{bmatrix} U_k(z_1) \\ U_k(z_2) \end{bmatrix} & \mathbf{I} \end{bmatrix}^\top}_{\text{update step}} \underbrace{\begin{bmatrix} 1 & \mathbf{O} \\ \begin{bmatrix} P_k(z_1) \\ P_k(z_2) \end{bmatrix} & \mathbf{I} \end{bmatrix}}_{\text{predict step}}. \tag{7}$$

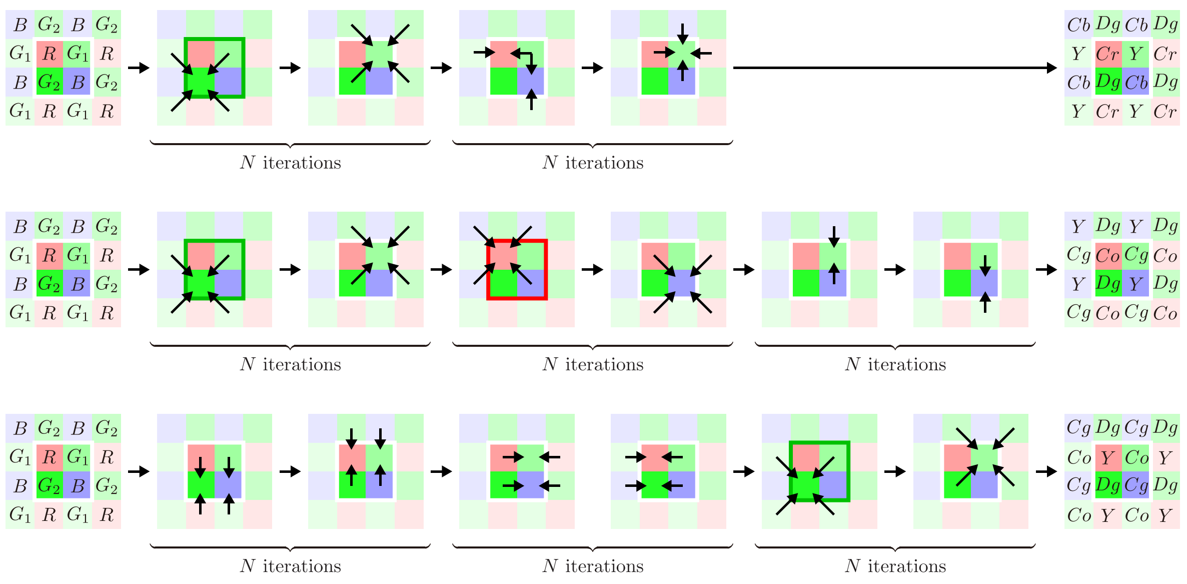

Fig. 2. Implementations of WSSTs (arrows, green-framed macropixels, and red-framed macropixels mean lifting step, $G1$-to-$G2$ prediction, and B-to-R prediction, respectively): (top-to-bottom) YDgCbCr-, YDgCoCg-, and YDgCoCg2-WSSTs.

$\mathbf{P}_j$ ($j \in \mathbb{N}$) is a 4×4 permutation matrix,

$$\mathbf{P}_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{P}_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{P}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{P}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$
$$\mathbf{P}_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{P}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

and $P_k(z_i)$, $U_k(z_i)$, $P_k(z_1, z_2)$, and $U_k(z_1, z_2)$ are polynomials with coefficients p_k and u_k:

$$P_k(z_i) = (1 + \overline{z}_i)p_k, \tag{9}$$
$$U_k(z_i) = (1 + z_i)u_k, \tag{10}$$
$$P_k(z_1, z_2) = \frac{1}{2}(1 + \overline{z}_1 + \overline{z}_2 + \overline{z}_1\overline{z}_2)p_k, \tag{11}$$
$$U_k(z_1, z_2) = \frac{1}{2}(1 + z_1 + z_2 + z_1 z_2)u_k. \tag{12}$$

Table I shows the coefficients p_k and u_k in the 5/3 wavelet transforms ($N = 1$) and 9/7 wavelet transforms ($N = 2$), and Fig. 2 shows implementations of the existing WSSTs. The pixels used in the predict and update steps are not limited to be within a macropixel; the surrounding pixels are also used to make the predict results more accurate. However, because they do not consider image features, their predictions may not be so accurate in some cases.

B. Star-Tetrix Transform

The STT, presented by Richter et al. [15], is a spectral-spatial transform for CFA-sampled raw image compression with JPEG XS. It consists of four steps based on 5/3 wavelet transforms. The first step generates chroma components C_b and C_r by predicting R and B from the four surrounding green components G^x ($x = \{l, r, t, b\}$), where l, r, t, and b respectively indicate the sample position to the left, right, top, and bottom of the current pixel, as

$$C_\text{b} = B - \left\lfloor \frac{G^l + G^r + G^t + G^b}{4} \right\rfloor, \tag{13}$$
$$C_\text{r} = R - \left\lfloor \frac{G^l + G^r + G^t + G^b}{4} \right\rfloor. \tag{14}$$

The second step generates the luma components Y_1 and Y_2 by updating G from the four surrounding C_b^xs and C_r^xs as

$$Y_1 = G + \left\lfloor \frac{C_\text{r}^l + C_\text{r}^r + C_\text{b}^t + C_\text{b}^b}{8} \right\rfloor, \tag{15}$$
$$Y_2 = G + \left\lfloor \frac{C_\text{r}^t + C_\text{r}^b + C_\text{b}^l + C_\text{b}^r}{8} \right\rfloor. \tag{16}$$

For simplicity, the white-balancing constants defined in [15] will not be considered in this study. The third step generates the luma difference component Δ by predicting Y_1 from the four surrounding Y_2s, as

$$\Delta = Y_1 - \left\lfloor \frac{Y_2^{l,t} + Y_2^{r,t} + Y_2^{l,b} + Y_2^{r,b}}{4} \right\rfloor. \tag{17}$$

The last step generates the final luma component Y by updating Y_2 from the four surrounding Δs, as

$$Y = Y_2 + \left\lfloor \frac{\Delta^{l,t} + \Delta^{r,t} + \Delta^{l,b} + \Delta^{r,b}}{8} \right\rfloor. \tag{18}$$

Although the STT here looks different from the WSSTs, we can extend (generalize) the STT to the WSSTs as described in the next section.

III. Extended Star-Tetrix Transforms and Edge-Aware Weighted 2-D Predict Steps

A. Type-I Extended Star-Tetrix Transforms: A New Wavelet-based Spectral-Spatial Transform

Definition: Although the original STT [15] consists of three 5/3 wavelet transforms as described above, we can extend it to a fourth version of the WSSTs (second version of the YDgCbCr-WSSTs). Since another type of extended STT (XSTT) will be introduced in the next subsection, we will refer to the XSTTs in this subsection as "type-I XSTTs" or "XSTT-Is" to distinguish between them. The XSTT-I $\mathfrak{S}_{\mathrm{I}}$ is represented as follows:

$$\mathfrak{S}_{\mathrm{I}} = \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\overline{z}_1, z_2) & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \prod_{k=N-1}^{0} \underbrace{\begin{bmatrix} \mathbf{I} & \boldsymbol{\mathcal{U}}_k \\ \mathbf{O} & \mathbf{I} \end{bmatrix}}_{\text{update step}} \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \boldsymbol{\mathcal{P}}_k & \mathbf{I} \end{bmatrix}}_{\text{predict step}}, \quad (19)$$

where

$$\boldsymbol{\mathcal{P}}_k = \frac{1}{2} \begin{bmatrix} P_k(z_2) & P_k(z_1) \\ P_k(\overline{z}_1) & P_k(\overline{z}_2) \end{bmatrix}, \ \boldsymbol{\mathcal{U}}_k = \frac{1}{2} \begin{bmatrix} U_k(z_2) & U_k(\overline{z}_1) \\ U_k(z_1) & U_k(\overline{z}_2) \end{bmatrix}. \quad (20)$$

Although we have omitted the white-balancing constants defined in [15], the parameters can be simply applied to $\boldsymbol{\mathcal{W}}_2(\overline{z}_1, z_2)$ in (19) if we desire it. The top of Fig. 4 shows an implementation of the XSTT-Is. The XSTT-Is will be further extended to edge-aware transforms later.

Remark-1: In the case of the 5/3 wavelet transforms, the first, second, and third steps in (19) represent the same process as (13)-(14), (15)-(16), and (17)-(18), respectively; i.e., the XSTT-I that uses 5/3 wavelet transforms is identical to the original STT. In addition, D_{g} in the XSTT-I is denoted as Δ in the original STT.

Remark-2: If there are no rounding operations in the lifting steps, we find that the XSTT-I that uses Haar transforms is identical to the YDgCbCr-WSST that uses Haar transforms:

$$\begin{aligned}\mathfrak{S}_{\mathrm{I}} &= \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ -1 & 1 & 0 & 0 \\ -1/2 & -1/2 & 1 & 0 \\ -1/2 & -1/2 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} 1 & 0 & 1/4 & 1/4 \\ 0 & 1 & 1/4 & 1/4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1/2 & -1/2 & 1 & 0 \\ -1/2 & -1/2 & 0 & 1 \end{bmatrix}. \quad (21)\end{aligned}$$

The implementation is shown at the top of Fig. 3. Consequently, the XSTT-Is can be considered to be a second version of the YDgCbCr-WSSTs. Note that the XSTT-I that uses Haar transforms outputs slightly different results from the YDgCbCr-WSST that uses Haar transforms due to the rounding operations on the lifting steps in the actual implementation.

B. Type-II Extended Star-Tetrix Transforms: A Simpler Version than Type-I Extended Star-Tetrix Transforms

Definition: Here, we describe another type of XSTT; called the "type-II XSTTs" or "XSTT-IIs." These are simpler than the XSTT-Is and a fifth version of the WSSTs (third version of the YDgCbCr-WSSTs). The XSTT-II $\mathfrak{S}_{\mathrm{II}}$ is represented as follows:

$$\mathfrak{S}_{\mathrm{II}} = \underbrace{\begin{bmatrix} \mathbf{I} & \widetilde{\boldsymbol{\mathcal{U}}}_0 \\ \mathbf{O} & \mathbf{I} \end{bmatrix}}_{\text{update step}} \begin{bmatrix} \boldsymbol{\mathcal{W}}_2(\overline{z}_1, z_2) & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \boldsymbol{\mathcal{P}}_0 & \mathbf{I} \end{bmatrix}}_{\text{predict step}}, \quad (22)$$

where

$$\widetilde{\boldsymbol{\mathcal{U}}}_0 = \frac{1}{2} \begin{bmatrix} U_0(z_2) & U_0(\overline{z}_1) \\ 0 & 0 \end{bmatrix}. \quad (23)$$

The XSTT-IIs are considered to be special cases of WSSTs because wavelet transforms with $N \geq 2$, such as 9/7 wavelet transforms, can only be applied to the second step in (22). The bottom of Fig. 4 shows an implementation of the XSTT-IIs. As with the XSTT-Is, they will be further extended to edge-aware transforms later.

Derivation: The XSTT-II in (22) is obtained with the following simple procedure. First, the XSTT-I that uses Haar transforms in (21) is rewritten as

$$\begin{aligned}\mathfrak{S}_{\mathrm{II}} &= \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ -1 & 1 & 0 & 0 \\ -1/2 & -1/2 & 1 & 0 \\ -1/2 & -1/2 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 1/4 & 1/4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1/2 & -1/2 & 1 & 0 \\ -1/2 & -1/2 & 0 & 1 \end{bmatrix}. \quad (24)\end{aligned}$$

Let $\mathfrak{S}_{\mathrm{II}}$ be the XSTT-II that uses Haar transforms. Next, in accordance with the extension method described in [14], the XSTT-II that uses Haar transforms in (24) can easily be extended to WSSTs in the form of (22).

Remark-1: The XSTT-IIs are simpler than the XSTT-Is because $\widetilde{\boldsymbol{\mathcal{U}}}_0$ in (23) does not require $U_0(z_1)$ or $U_0(\overline{z}_2)$ belonging to the bottom row of $\boldsymbol{\mathcal{U}}_0$ in (20), i.e., $U_0(z_1) = U_0(\overline{z}_2) = 0$.

Remark-2: Like the XSTT-I, if there are no rounding operations in the lifting steps, the XSTT-II that uses Haar transforms is identical to the YDgCbCr-WSST that uses Haar transforms, as shown in (24). The implementation is shown at the bottom of Fig. 3. Consequently, the XSTT-IIs can be considered to be a third version of the YDgCbCr-WSSTs. Note that the XSTT-II that uses Haar transforms outputs slightly different results from the XSTT-I and YDgCbCr-WSST that use Haar transforms due to the rounding operations.

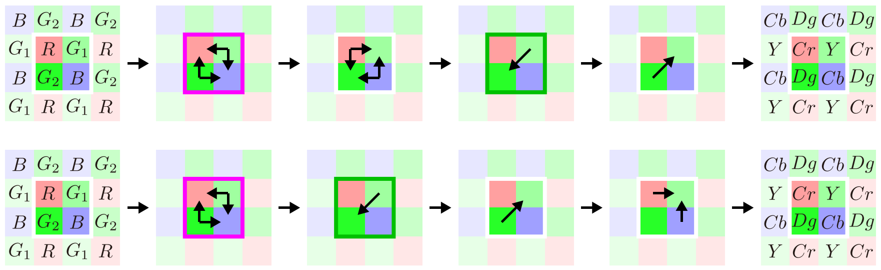

Fig. 3. Implementations of XSTTs that use Haar transforms (arrows, green-framed macropixels, and magenta-framed macropixels mean lifting step, $G1$-to-$G2$ prediction, and (G_1, G_2)-to-R and (G_1, G_2)-to-B predictions, respectively): (top) XSTT-Is and (bottom) XSTT-IIs.

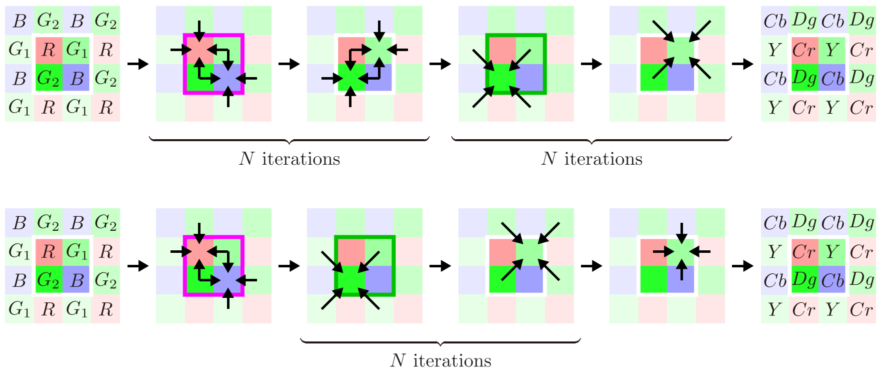

Fig. 4. Implementations of XSTTs (arrows, green-framed macropixels, and magenta-framed macropixels mean lifting step, $G1$-to-$G2$ prediction, and (G_1, G_2)-to-R and (G_1, G_2)-to-B predictions, respectively): (top) XSTT-Is and (bottom) XSTT-IIs.

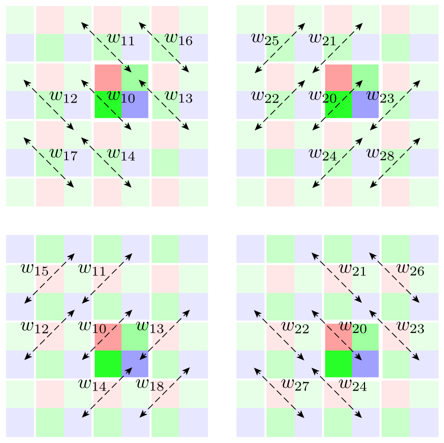
w11 w16 w12 w10 w13 w17 w14
w25 w21 w22 w20 w23 w24 w28
w15 w11 w12 w10 w13 w14 w18
w21 w26 w22 w20 w23 w27 w24

Fig. 5. Weights for the 2-D diagonal predict step (dashed arrows mean to calculate the difference between two pixels): (top) for G_1-to-G_2 prediction and (bottom) B-to-R prediction.

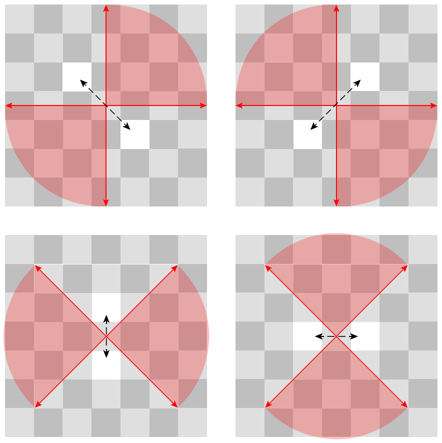

Fig. 6. Relation between weight calculation and edge direction detection (dashed arrows mean the difference between two pixels and the red areas are the detectable edge direction ranges): (top-left) left diagonal weight, (top-right) right diagonal weight, (bottom-left) vertical weight, and (bottom-right) horizontal weight.

C. Edge-Aware Weighted 2-D Diagonal Predict Steps

In this subsection, we extend the polynomial $P_k(z_1, z_2)$ in (11), which is in the 2-D *diagonal* predict step of the wavelet transform $\boldsymbol{\mathcal{W}}_2(z_1, z_2)$ in (6), to

$$\widetilde{P}_k(z_1, z_2) = \frac{W_1(1 + \bar{z}_1\bar{z}_2) + W_2(\bar{z}_1 + \bar{z}_2)}{W_1 + W_2} p_k, \tag{25}$$

where W_m $(m = 1, 2)$ is a weight,

$$W_m = \sum_n w_{mn} + \varepsilon, \tag{26}$$

w_{mn} is automatically determined in accordance with the image features as described later, and an extremely small value ε is added to (26) to avoid division by zero. Hence, the original 2-D diagonal predict step is divided into two 1-D diagonal (left- and right-diagonal) predict steps and they are weighted to consider the image features. When $W_1 = W_2 = 1$, it is clear that $\widetilde{P}_k(z_1, z_2) = P_k(z_1, z_2)$. Consequently, for the cases of G_1-to-G_2 and B-to-R predictions, we can rewrite $\widetilde{P}_k(z_1, z_2)$

in (25), as

$$\widetilde{P}_k(\overline{z}_1, z_2) = \frac{W_1(1 + z_1\overline{z}_2) + W_2(z_1 + \overline{z}_2)}{W_1 + W_2} p_k \quad \text{for } G_1\text{-to-}G_2 \text{ prediction}, \tag{27}$$

$$\widetilde{P}_k(\overline{z}_1, \overline{z}_2) = \frac{W_1(1 + z_1 z_2) + W_2(z_1 + z_2)}{W_1 + W_2} p_k \quad \text{for } B\text{-to-}R \text{ prediction}, \tag{28}$$

and define w_{1n} and w_{2n} as

$$w_{1n} = \begin{cases} |Z_n(z_1 - \overline{z}_2)G_1|^\gamma & \text{for } G_1\text{-to-}G_2 \text{ prediction} \\ |Z_n(z_1 - z_2)B|^\gamma & \text{for } B\text{-to-}R \text{ prediction} \end{cases}, \tag{29}$$

$$w_{2n} = \begin{cases} |Z_n(1 - z_1\overline{z}_2)G_1|^\gamma & \text{for } G_1\text{-to-}G_2 \text{ prediction} \\ |Z_n(1 - z_1 z_2)B|^\gamma & \text{for } B\text{-to-}R \text{ prediction} \end{cases}, \tag{30}$$

where $\gamma \in \mathbb{R}$ is an arbitrary adjustment parameter that evaluates the edge-likeness ($\widetilde{P}_k(z_1, z_2) = P_k(z_1, z_2)$ if $\gamma = 0$) and Z_n is a delay:

$$Z_n = \begin{cases} 1 & \text{if } n = 0 \text{ (center)} \\ z_2 & \text{if } n = 1 \text{ (top)} \\ z_1 & \text{if } n = 2 \text{ (left)} \\ \overline{z}_1 & \text{if } n = 3 \text{ (right)} \\ \overline{z}_2 & \text{if } n = 4 \text{ (bottom)} \\ z_1 z_2 & \text{if } n = 5 \text{ (top-left)} \\ \overline{z}_1 z_2 & \text{if } n = 6 \text{ (top-right)} \\ z_1 \overline{z}_2 & \text{if } n = 7 \text{ (bottom-left)} \\ \overline{z}_1 \overline{z}_2 & \text{if } n = 8 \text{ (bottom-right)} \end{cases}. \tag{31}$$

Figure 5 overviews the weights for the edge-aware weighted 2-D diagonal predict steps. γ was empirically set to $\gamma = 1$ in the lossless compression experiment and $\gamma = 1/2$ in the lossy compression experiment. Since the edge-aware weighted 2-D diagonal predict steps can be applied to any $P_k(z_1, z_2)$ in (6), we can extend all of the WSSTs and XSTTs to efficiently decorrelate the CFA-sampled raw images while taking the edge information into account. Also, to avoid extra bits and a significant amount of complexity, the weights are not transmitted to the decoder and are re-calculated in the decoder. Note that, as in Section IV-C, the weights between the encoder and decoder tend to be more different at higher lossy compression levels.

D. Edge-Aware Weighted 2-D Horizontal-Vertical Predict Steps

Like the edge-aware weighted 2-D diagonal predict steps in (25), we can also extend the polynomial $P_k(z_i)$ in (9), which is in the 2-D *horizontal-vertical* predict step in (23), as follows:

$$\widetilde{P}_k(z_i) = \frac{W_i}{W_1 + W_2}(1 + \overline{z}_i)p_k. \tag{32}$$

When $W_1 = W_2 = 1$, it is clear that $\widetilde{P}_k(z_i) = P_k(z_i)$. Naturally, we can rewrite $\widetilde{P}_k(z_i)$ in (32) as

$$\widetilde{P}_k(\overline{z}_i) = \frac{W_i}{W_1 + W_2}(1 + z_i)p_k. \tag{33}$$

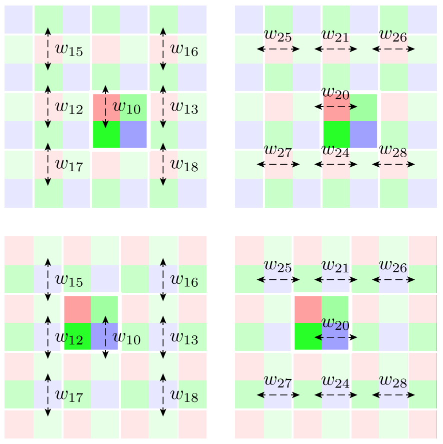

Fig. 7. Weights for the 2-D horizontal-vertical predict steps (dashed arrows mean to calculate the difference between two pixels): (top) (G_1, G_2)-to-R prediction and (bottom) (G_1, G_2)-to-B prediction.

Moreover, we can rewrite w_{mn} as

$$w_{1n} = \begin{cases} |Z_n(1 - z_2)G_2|^\gamma & \text{for } (G_1, G_2)\text{-to-}R \text{ prediction} \\ |Z_n(1 - \overline{z}_2)G_1|^\gamma & \text{for } (G_1, G_2)\text{-to-}B \text{ prediction} \end{cases}, \tag{34}$$

$$w_{2n} = \begin{cases} |Z_n(1 - z_1)G_1|^\gamma & \text{for } (G_1, G_2)\text{-to-}R \text{ prediction} \\ |Z_n(1 - \overline{z}_1)G_2|^\gamma & \text{for } (G_1, G_2)\text{-to-}B \text{ prediction} \end{cases}, \tag{35}$$

where γ and Z_n are the same as in the case of the edge-aware weighted 2-D diagonal predict steps. Figure 7 overviews the weights for the 2-D horizontal-vertical predict steps. Since the vertical (horizontal) weight w_{mn} detects the horizontal (vertical) edge direction with a 90° range, the top and bottom (left and right) weights are omitted. Like the edge-aware weighted 2-D diagonal predict steps, γ was empirically set as $\gamma = 1$ in the lossless compression experiment and $\gamma = 1/2$ in the lossy compression experiment. Since the edge-aware weighted 2-D horizontal-vertical predict steps can be applied to only $P_k(z_i)$ in (20), unlike the diagonal case, and $P_k(z_i)$ in (20) is included only in the XSTTs, we will be able to extend only the XSTTs. Also, as in the diagonal case, the weights are re-calculated in the decoder.

IV. Experimental Results

A. Preparation

We compared our XSTTs and EXSTTs, which are the XSTTs that use the edge-aware weighted 2-D predict steps, with existing spectral-spatial transforms including simple ones which do not apply white balance or gamma correction to themselves for CAMRA [12], the WSSTs [14], the STT [15] which do not apply white-balancing constants to themselves, and the weighted WSSTs (WWSSTs) [19].[1] For reference, we

[1]The YDgCoCg-WSSTs/WWSSTs performed the best among the three types of WSSTs/WWSSTs in preliminary experiments, so the comparisons presented here show only YDgCoCg-WSSTs/WWSSTs; hereafter, the YDgCoCg-WSSTs/WWSSTs will simply be referred to as "WSSTs/WWSSTs." In addition, the weights of WWSST were updated to the ones in this study.

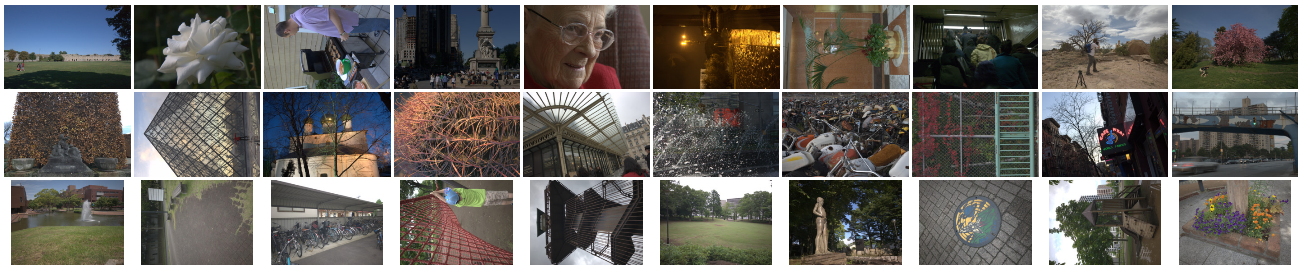

Fig. 8. Test images developed from raw data by a simple camera processing pipeline: (top) general images (*#0500*, *#1000*, *#1500*, *#2000*, *#2500*, *#3000*, *#3500*, *#4000*, *#4500*, and *#5000*), (middle) images with many edges (*#0092*, *#0482*, *#0548*, *#1137*, *#1145*, *#1463*, *#2239*, *#2912*, *#3394*, and *#3419*), and (bottom) mobile phone images (*m01*, *m02*, *m03*, *m04*, *m05*, *m06*, *m07*, *m08*, *m09*, and *m10*).

compared them with direct JPEG 2000 compression of full-color RGB images obtained by a simple camera processing pipeline (black level correction, white balance, demosaicing [1], and gamma correction), as shown in [20]. The 5/3 and 9/7 wavelet transforms[2] were used in the lossless and lossy compressions based on the spectral-spatial transforms; the XSTT-II and EXSTT-II have 5/3 wavelet transforms for the first and last steps even for lossy compression because of their special structures. Note that the XSTT-I that uses 5/3 wavelet transforms is identical to the STT and the one that uses 9/7 wavelet transforms can be regarded as an extended version of the STT, as indicated above. The transformed images were adjusted to positive values and compressed with JPEG 2000 [6][3] by using 'imwrite.m' with the default settings (except for compression ratio) in MATLAB. To investigate the effect on high-quality camera images, we used 20 DNG data, whose bit depths are 12 or 14 bits, in the MIT-Adobe FiveK dataset [21] after loading them via 'rawread.m' in MATLAB and forming them into an "RGGB" array. The high-quality camera images included ten general images used in [14] and ten intuitively selected images that seemed to have many edges. To investigate the effect on mobile phone images, which are noisier than the images in the MIT-Adobe FiveK dataset, we used ten DNG data, whose bit depths are 12 bits, acquired with an iPhone SE (3rd generation). Figure 8 shows the 30 test images. In addition, we used the lossless bitrates (LBRs) [bpp] in lossless compression and the Bjøntegaard delta peak signal-to-noise ratios (BD-PSNRs) [dB] between about 2–5 bpp in lossy compression for a fair comparison. We selected the macropixel spectral-spatial transform (MSST) [10] as the basis for the comparison using the BD-PSNRs and computed the BD-PSNRs after the CFA-sampled images were developed into full-color RGB images by using a simple camera processing pipeline similar to that used for direct JPEG 2000 compression.

B. Decorrelation of CFA-Sampled Raw Images

Figure 9 shows only the decorrelated (transformed) subband images with the XSTT-I and EXSTT-I that use 9/7 wavelet transforms because the differences depending on the transforms were not clear. The grayscale-transformed images were directly compressed with JPEG 2000.

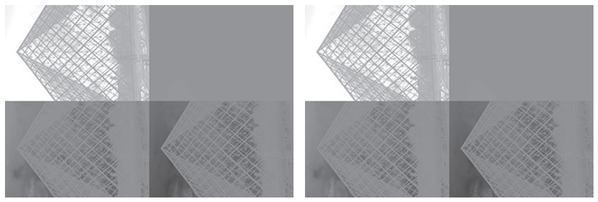

Fig. 9. Decorrelated subband images of *#0482* (clockwise from the top-left: Y, $D_{\rm g}$, C_2, and C_1): (left) XSTT-I and (right) EXSTT-I that use 9/7 wavelet transforms.

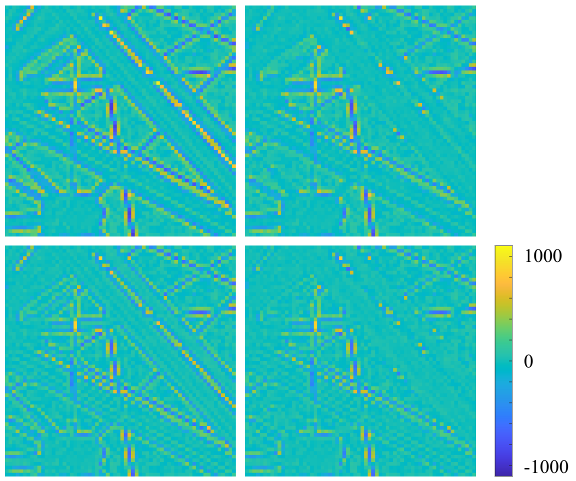

Fig. 10. Part of $D_{\rm g}$ components of *#0482* represented in pseudo color: (top) XSTT-I and EXSTT-I that use 5/3 wavelet transforms, (bottom) XSTT-I and EXSTT-I that use 9/7 wavelet transforms.

To clearly show the effect of the edge-aware weighted 2-D predict steps, Fig. 10 and Table II show parts of the $D_{\rm g}$ components, the high-frequency information between the G_1 and G_2 components, of the images transformed by the XSTTs/EXSTTs and the reduction (improvement) in mean squared error (MSE)[4] [%] of the Dg components after applying the edge-aware weighted 2-D predict steps to the XSTTs. Since the $D_{\rm g}$ components in the XSTT-II/EXSTT-IIs look almost identical to the ones in the XSTT-I/EXSTT-Is, we omitted the XSTT-II/EXSTT-IIs from Fig. 10. The

[2]We did not apply rounding operations to the 9/7 wavelet transforms for better lossy compression.

[3]JPEG 2000 that uses the wavelet transforms has high affinity to any spectral-spatial transform. Of course, other codecs, such as JPEG XR and JPEG XS, can be applied to any of the spectral-spatial transforms.

[4]Strictly speaking, these values are not MSEs. However, we indicate them as such because $D_{\rm g}$ is like the error between the G_1 and G_2 components.

TABLE II
IMPROVEMENT IN MSES [%] OF D_{G} COMPONENTS AFTER INCORPORATING EDGE-AWARE WEIGHTED 2-D PREDICT STEPS IN THE XSTTS (LOWER MSES ARE BETTER).

	type-I		type-II			type-I		type-II			type-I		type-II	
	5/3	9/7	5/3	9/7		5/3	9/7	5/3	9/7		5/3	9/7	5/3	9/7
#0500	+1.42	+1.51	+1.19	+1.37	*#0092*	−16.17	−12.38	−14.72	−11.48	*m01*	−9.21	−5.71	−8.43	−5.01
#1000	+1.24	+1.40	+0.36	+1.27	*#0482*	−41.66	−34.05	−39.25	−31.76	*m02*	−6.66	−3.64	−6.18	−3.17
#1500	−17.00	−10.91	−13.59	−9.29	*#0548*	−17.04	−10.95	−14.85	−9.83	*m03*	−20.58	−16.22	−15.78	−12.72
#2000	−13.17	−8.86	−9.03	−5.49	*#1137*	−31.92	−23.62	−30.20	−22.74	*m04*	−17.42	−11.34	−16.59	−10.83
#2500	+0.66	+0.68	+0.41	+0.73	*#1145*	−26.42	−19.31	−26.14	−18.68	*m05*	−29.04	−23.56	−24.50	−19.60
#3000	−4.89	−2.78	−4.29	−1.99	*#1463*	−17.91	−14.20	−17.35	−14.52	*m06*	−11.40	−7.72	−10.24	−6.94
#3500	−16.09	−7.09	−14.83	−5.91	*#2239*	−26.81	−19.08	−25.11	−18.24	*m07*	−10.03	−6.22	−9.34	−5.72
#4000	−8.22	−8.53	−1.82	−1.17	*#2912*	−25.99	−12.83	−24.51	−10.31	*m08*	−7.87	−3.79	−8.13	−4.16
#4500	−13.33	−9.25	−11.80	−7.82	*#3394*	−25.95	−20.36	−23.04	−18.22	*m09*	−22.69	−18.72	−19.96	−16.67
#5000	−6.67	−3.32	−6.59	−3.25	*#3419*	−41.44	−38.39	−38.31	−36.41	*m10*	−9.25	−4.89	−9.22	−5.07
Avg.	−9.84	−7.66	−5.92	−3.38	Avg.	−30.08	−24.82	−27.81	−23.23	Avg.	−14.47	−10.26	−12.81	−8.97

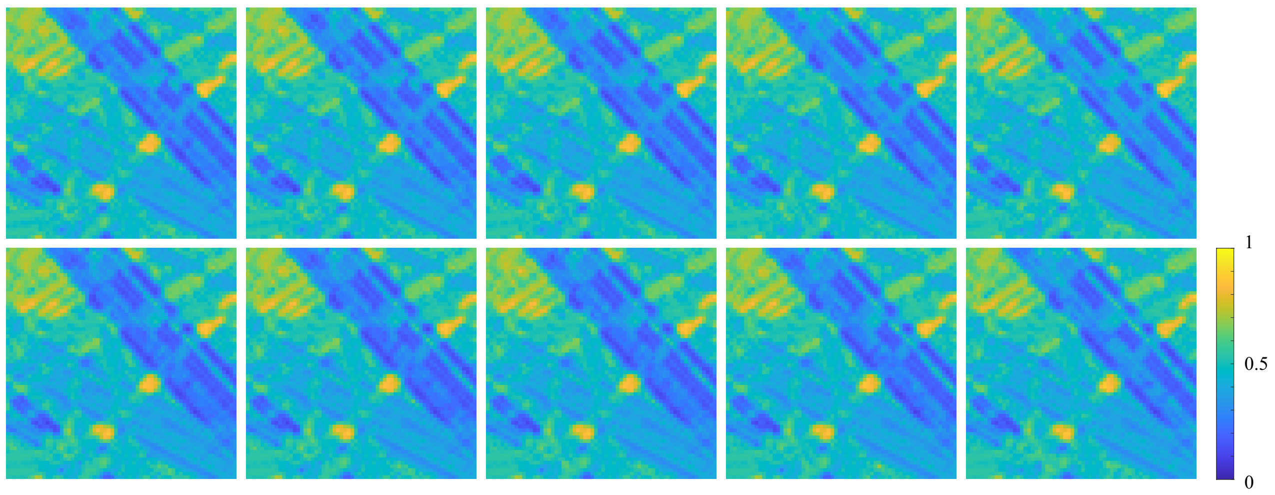

Fig. 11. Weights for $G1$-to-$G2$ predictions, which generate D_{g} components, in the EXSTT-Is for part of *#0482* represented in pseudo color: (top) $W_1/(W_1 + W_2)$ in 5/3 wavelet transforms, (bottom) the first $W_1/(W_1 + W_2)$ in 9/7 wavelet transforms, (first column) ideal weights on the encoder, and (second-to-last columns) lossy weights on the decoder when 5, 4, 3, and 2 bpp.

EXSTTs reduced the average MSEs of the D_{g} components by about 3.38–9.84 % for general images, 23.23–30.08 % for images with many edges, and 8.97–14.47 % for mobile phone images. Figure 11 shows the corresponding weights for the part in Fig. 10. We omitted the second weights in the 9/7 wavelet transforms, which looked almost identical to the first weights. Figures 10 and 11 show that the weights assigned along the edge directions helped to reduce the D_{g} energies. There were also slight differences in weight between the 5/3 and 9/7 wavelet transforms. On the other hand, unfortunately, the weights between the encoder and decoder tended to be more different for higher lossy compression levels. Since the differences may affect coding performance, we will investigate this finding in Section IV-C.

C. CFA-Sampled Raw Image Compression

1) Effect of Edge-Aware Weighted 2-D Predict Steps: Tables III and IV show the improvements in the lossless and lossy compressions for the CFA-sampled raw images after incorporating the edge-aware weighted 2-D predict steps in the XSTTs: the differences in LBR and BD-PSNR were obtained by subtracting the LBR and BD-PSNR of the original transforms from those of the edge-aware transforms in lossless and lossy compression, respectively. For lossless compression, although all EXSTTs hardly showed any differences on the general and mobile phone images, they gave the best results in the case of images with many edges. In contrast, for lossy compression, the EXSTTs gave worse results than the XSTTs because we re-calculated *lossy* weights on the decoder (Fig. 11). If we would like to use the *ideal* weights on the decoder, the encoder-calculated weights must be transmitted to the decoder together with a lot of side information and it is more practical to use the XSTTs instead of the EXSTTs for lossy compression.

2) Comparisons with Other Methods: Table V, Table VI, and Fig. 12 show the LBRs in lossless compression, BD-PSNRs in lossy compression, and rate-distortion (R-D) curves in lossy compression, respectively. For lossless compression, the compression-first approaches using the spectral-spatial transforms clearly outperformed the direct JPEG 2000 compression of full-color RGB images, because the full-color RGB images had three times the components compared with a CFA-sampled raw image. For lossy compression, the compression-first approaches using the spectral-spatial transforms outperformed the direct JPEG 2000 compression at higher bitrates (4 bits or more), but not at lower bitrates. For high-quality

TABLE III
IMPROVEMENT IN LBRS [BPP] FOR LOSSLESS COMPRESSION AFTER INCORPORATING EDGE-AWARE WEIGHTED 2-D PREDICT STEPS IN THE XSTTS (LOWER LBRS ARE BETTER).

	type-I	type-II		type-I	type-II		type-I	type-II
#0500	+0.01	+0.01	*#0092*	−0.02	−0.02	*m01*	±0.00	±0.00
#1000	+0.01	+0.01	*#0482*	−0.08	−0.08	*m02*	±0.00	±0.00
#1500	±0.00	±0.00	*#0548*	−0.02	−0.02	*m03*	−0.01	−0.01
#2000	+0.01	+0.01	*#1137*	−0.02	−0.02	*m04*	−0.01	−0.01
#2500	+0.01	+0.01	*#1145*	−0.02	−0.02	*m05*	−0.01	−0.01
#3000	±0.00	±0.00	*#1463*	−0.02	−0.02	*m06*	−0.01	−0.01
#3500	±0.00	±0.00	*#2239*	−0.02	−0.02	*m07*	−0.01	−0.01
#4000	+0.01	+0.01	*#2912*	−0.04	−0.04	*m08*	−0.01	−0.01
#4500	±0.00	±0.00	*#3394*	−0.02	−0.02	*m09*	−0.01	−0.01
#5000	±0.00	±0.00	*#3419*	−0.02	−0.02	*m10*	−0.01	−0.01
Avg.	±0.00	±0.00	Avg.	−0.03	−0.03	Avg.	−0.01	−0.01

TABLE IV
IMPROVEMENT IN BD-PSNRS [DB] FOR LOSSY COMPRESSION AFTER INCORPORATING EDGE-AWARE WEIGHTED 2-D PREDICT STEPS IN THE XSTTS (HIGHER BD-PSNRS ARE BETTER, AND THE VALUES IN () INDICATE THE IDEAL WEIGHTS WITHOUT SIDE INFORMATION CONSIDERED).

	type-I		type-II			type-I		type-II			type-I		type-II	
#0500	−0.27	(−0.06)	−0.15	(−0.07)	*#0092*	−0.62	(+0.10)	−1.54	(+0.05)	*m01*	−0.29	(−0.04)	−0.09	(−0.04)
#1000	−0.36	(−0.11)	−0.15	(−0.05)	*#0482*	−0.24	(+0.19)	−0.08	(+0.25)	*m02*	−0.11	(±0.00)	−0.07	(−0.02)
#1500	−0.42	(−0.03)	−0.78	(−0.03)	*#0548*	−0.06	(±0.00)	±0.00	(+0.02)	*m03*	−1.59	(−0.15)	−1.39	(−0.01)
#2000	−0.30	(−0.08)	−0.09	(−0.01)	*#1137*	−0.08	(+0.01)	+0.01	(+0.05)	*m04*	−0.15	(−0.05)	−2.82	(+0.01)
#2500	−0.33	(−0.11)	−0.15	(−0.06)	*#1145*	−0.06	(+0.03)	+0.01	(+0.07)	*m05*	−0.10	(±0.00)	−0.41	(+0.01)
#3000	−0.82	(−0.04)	−0.64	(+0.01)	*#1463*	−0.21	(+0.07)	−0.12	(+0.09)	*m06*	−0.13	(+0.05)	−1.38	(+0.01)
#3500	−0.52	(−0.11)	−0.10	(−0.05)	*#2239*	−0.03	(+0.05)	−0.04	(+0.03)	*m07*	−0.07	(±0.00	−0.07	(−0.01)
#4000	−0.78	(−0.10)	−0.23	(−0.04)	*#2912*	−0.03	(+0.08)	+0.04	(+0.09)	*m08*	−0.09	(−0.01)	−0.06	(−0.03)
#4500	−0.15	(−0.02)	−0.06	(±0.00)	*#3394*	−2.13	(+0.03)	−0.51	(+0.04)	*m09*	−0.25	(±0.00	−1.16	(−0.02)
#5000	−0.16	(−0.04)	−0.07	(−0.02)	*#3419*	−0.67	(−0.01)	−0.09	(+0.08)	*m10*	−0.04	(+0.04)	+0.01	(+0.05)
Avg.	−0.41	(−0.07)	−0.24	(−0.03)	Avg.	−0.41	(+0.06)	−0.23	(+0.08)	Avg.	−0.28	(−0.01)	−0.74	(−0.01)

camera images, the XSTTs/EXSTTs produced results equal to or better than the conventional spectral-spatial transforms: especially for images with many edges, the EXSTT-I improved them by about 0.03–0.19 bpp in average LBR and the XSTTs improved them by about 0.16–0.96 dB in average BD-PSNR. In addition, for mobile phone images, i.e. noisy images, the WSSTs/WWSSTs performed the best, whereas the XSTTs/EXSTTs showed similar tends to the case of high-quality camera images. Also, as shown in Fig. 13, there was no noticeable problem with the images even when they had been compressed with 2 bpp and developed with the simple camera processing pipeline.

V. CONCLUSION

This study described two types of XSTT and their edge-aware versions, EXSTTs, with no side information and little extra complexity for CFA-sampled raw image compression. They were obtained by (i) extending the STT, which is one of the latest spectral-spatial transforms, to a new version of our previously proposed WSSTs and a simpler version, (ii) considering that each 2-D predict step of the wavelet transforms is a combination of two 1-D diagonal or horizontal-vertical transforms, and (iii) weighting the transforms along the edge directions in the images. Compared with XSTTs, the EXSTTs decorrelated the images well. For images with many edges, our XSTTs/EXSTTs produced results better than the conventional methods in lossless and lossy compressions without reducing the compression efficiency for general images. For mobile phone images, our previous work performed the best, whereas the XSTTs/EXSTTs showed similar trends to the case of high-quality camera images.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and Dr. K. Shirai for providing many constructive suggestions that significantly improved the presentation of this paper.

REFERENCES

[1] H. S. Malvar, L. He, and R. Cutler, "High quality linear interpolation for demosaicing of Bayer-patterned color images," in *Proc. ICASSP*, Montreal, Quebec, Canada, May 2004, pp. 485–488.

[2] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Beyond color difference: Residual interpolation for color image demosaicking," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1288–1300, Mar. 2016.

[3] D. S. Tan, W.-Y. Chen, and K.-L. Hua, "DeepDemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 5, pp. 2408–2419, May 2018.

[4] Y. Wang, S. Yin, S. Zhu, Z. Ma, R. Xiong, and B. Zeng, "NTSDCN: New three-stage deep convolutional image demosaicking network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 3725–3729, Nov. 2020.

[5] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electr.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[6] A. Skodras, C. Christopoulis, and T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[7] A. Descampe, T. Richter, T. Ebrahimi, S. Fößel, J. Keinert, T. Bruylants, P. Pellegrin, C. Buysschaert, and G. Rouvroy, "JPEG XS–A new standard for visually lossless low-latency lightweight image coding," *Proc. IEEE*, vol. 109, no. 9, pp. 1559–1577, May 2021.

[8] "Kodak Lossless True Color Image Suite," *Kodak [Online]*, Available: http://r0k.us/graphics/kodak/.

[9] N. Zhang and X. Wu, "Lossless compression of color mosaic images," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1379–1388, Jun. 2006.

[10] H. S. Malvar and G. J. Sullivan, "Progressive-to-lossless compression of color-filter-array images using macropixel spectral-spatial transformation," in *Proc. DCC*, Snowbird, UT, Apr. 2012, pp. 3–12.

TABLE V
LBRs [BPP] IN LOSSLESS COMPRESSION.

	Direct	MSST [10]	CAMRA [12]	WSST [14]	STT [15]	WWSST [19]	XSTT-I	XSTT-II	EXSTT-I	EXSTT-II
#0500	(37.56)	6.43	6.41	**6.39**	6.40	6.40	6.40	6.42	6.41	6.43
#1000	(34.84)	6.38	6.40	**6.37**	6.40	6.38	6.40	6.41	6.41	6.42
#1500	(33.39)	8.44	8.38	**8.32**	**8.32**	**8.32**	**8.32**	8.34	**8.32**	8.34
#2000	(34.14)	5.27	5.25	5.23	**5.19**	5.22	**5.19**	5.20	**5.19**	5.21
#2500	(36.24)	6.27	6.31	**6.25**	6.28	**6.25**	6.28	6.30	6.29	6.30
#3000	(37.94)	5.98	5.97	5.92	**5.91**	5.93	**5.91**	5.92	**5.91**	5.92
#3500	(29.80)	7.46	7.42	7.36	**7.35**	7.36	**7.35**	7.37	**7.35**	7.37
#4000	(39.72)	**8.62**	8.66	**8.62**	8.65	**8.62**	8.65	8.66	8.66	8.67
#4500	(34.41)	7.12	7.04	7.02	6.94	7.00	6.94	6.95	**6.93**	6.94
#5000	(35.30)	6.14	6.11	**5.98**	6.01	**5.98**	6.01	6.03	6.01	6.03
Avg.	(35.33)	6.81	6.80	**6.75**	**6.75**	**6.75**	**6.75**	6.76	**6.75**	6.76
#0092	(34.03)	7.21	7.15	7.05	6.88	7.01	6.88	6.85	6.86	**6.83**
#0482	(34.17)	9.19	9.07	9.05	9.01	8.95	9.01	9.01	**8.93**	**8.93**
#0548	(35.69)	6.98	6.81	6.84	6.77	6.83	6.77	6.78	**6.75**	6.76
#1137	(34.06)	8.31	8.22	8.22	8.17	8.19	8.17	8.18	**8.15**	8.16
#1145	(34.02)	8.96	8.93	8.85	8.83	8.82	8.83	8.84	**8.81**	**8.81**
#1463	(37.46)	**7.46**	7.58	7.53	7.48	7.51	7.48	7.48	**7.46**	7.47
#2239	(33.99)	6.30	6.23	6.21	6.15	6.19	6.15	6.17	**6.13**	6.15
#2912	(34.04)	6.16	6.06	5.99	5.96	5.95	5.96	5.98	**5.91**	5.93
#3394	(36.00)	8.50	8.45	8.40	8.37	8.39	8.37	8.37	**8.36**	**8.36**
#3419	(32.11)	8.29	8.19	8.15	8.08	8.13	8.08	8.09	**8.06**	8.07
Avg.	(34.56)	7.73	7.67	7.63	7.57	7.60	7.57	7.58	**7.54**	7.55
m01	(36.72)	7.56	7.49	7.44	7.41	7.42	7.41	7.43	**7.40**	7.43
m02	(35.87)	7.44	7.36	7.32	**7.27**	7.30	**7.27**	7.30	**7.27**	7.29
m03	(30.83)	6.61	6.69	6.50	6.53	**6.48**	6.53	6.49	6.53	**6.48**
m04	(35.27)	7.10	6.99	6.96	6.94	6.94	6.94	6.96	**6.93**	6.95
m05	(34.61)	6.23	6.21	6.16	6.19	**6.13**	6.19	6.21	6.17	6.19
m06	(31.71)	7.13	7.28	6.92	7.00	6.90	7.00	6.90	6.99	**6.89**
m07	(36.96)	6.45	6.38	6.30	6.30	**6.28**	6.30	6.33	6.29	6.32
m08	(36.98)	8.35	8.21	8.14	**8.09**	8.12	**8.09**	8.12	**8.09**	8.11
m09	(31.36)	6.58	6.74	6.44	6.57	**6.42**	6.57	6.45	6.56	6.44
m10	(36.23)	7.75	7.64	7.57	7.55	7.55	7.55	7.57	**7.54**	7.56
Avg.	(34.65)	7.12	7.10	6.98	6.99	**6.96**	6.99	6.98	6.98	6.97

[11] M. Hernández-Cabronero, M. W. Marcellin, I. Blanes, and J. Serra-Sagristà, "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.

[12] 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.

[13] 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, Jan. 2020.

[14] T. Suzuki, "Wavelet-based spectral-spatial transforms for CFA-sampled raw camera image compression," *IEEE Trans. Image Process.*, vol. 29, no. 1, pp. 433–444, Dec. 2020.

[15] T. Richter, S. Fößel, A. Descampe, and G. Rouvroy, "Bayer pattern compression with JPEG XS," *IEEE Trans. Image Process.*, vol. 30, pp. 6557–6569, Jul. 2021.

[16] W. Ding, F. Wu, X. Wu, and S. Li, "Adaptive directional lifting-based wavelet transform for image coding," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 416–427, Feb. 2007.

[17] Y. Tanaka, M. Hasegawa, S. Kato, and T. Q. Nguyen, "Adaptive directional wavelet transform based on directional prefiltering," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 934–945, Apr. 2010.

[18] T. Yoshida, T. Suzuki, S. Kyochi, and M. Ikehara, "Two dimensional non-separable adaptive lifting structure of discrete wavelet transform," *IEICE Trans. Fundamentals*, vol. E94-A, no. 10, pp. 1920–1927, Oct. 2011.

[19] L. Huang and T. Suzuki, "Weighted wavelet-based spectral-spatial transforms for CFA-sampled raw image compression considering image features," in *Proc. ICASSP*, Singapore (Hybrid), May 2022, pp. 1850–1854.

[20] MathWorks Help Center, "Implement Digital Camera Processing Pipeline," https://jp.mathworks.com/help/images/end-to-end-implementation-of-digital-camera-processing-pipeline.html, (Accessed on 06/06/2022).

[21] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *Proc. CVPR*, Colorado Springs, CO, Jun. 2011, pp. 97–104.

Taizo Suzuki (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Japan, in 2004, 2006, and 2010, respectively. From 2006 to 2008, he was with Toppan Printing Company, Ltd., Japan. From 2008 to 2011, he was a Research Associate of the Global Center of Excellence (G-COE), Keio University, Japan. From 2010 to 2011, he was a Research Fellow of the Japan Society for the Promotion of Science (JSPS) and a Visiting Scholar at the Video Processing Group, University of California at San Diego, La Jolla, CA, USA. From 2011 to 2012, he was an Assistant Professor with Nihon University, Japan. In 2012, he joined the University of Tsukuba, Japan as an Assistant Professor, where he has been an Associate Professor since 2019. His current research interests include signal processing and its application to media contents. From 2017 to 2021, he was an Associate Editor of the *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*.

Liping Huang (Student Member, IEEE) received a B.Eng. degree in electrical and information engineering from Jimei University, China, in 2019. In 2019, she joined the University of Tsukuba, Japan as a Research Student, and since 2021, she has been in the Master's Program in Computer Science. Her current research interest is image and video processing.

TABLE VI
BD-PSNRs [dB] in lossy compression.

	Direct	MSST [10]	CAMRA [12]	WSST [14]	STT [15]	WWSST [19]	XSTT-I	XSTT-II	EXSTT-I	EXSTT-II
#0500	(0.75)	–	0.37	1.38	1.35	1.19	**1.48**	1.24	1.22	1.09
#1000	(0.03)	–	0.31	1.31	1.07	1.11	**1.35**	1.02	0.99	0.88
#1500	(2.33)	–	0.91	**1.49**	1.43	0.65	**1.49**	1.40	1.08	0.62
#2000	(0.52)	–	0.25	1.17	1.18	1.09	**1.39**	1.27	1.09	1.18
#2500	(−3.13)	–	0.21	**1.23**	0.67	1.08	0.91	0.58	0.57	0.43
#3000	(−2.46)	–	0.53	1.34	1.24	0.00	**1.43**	1.25	0.62	0.61
#3500	(−1.02)	–	0.71	2.01	1.76	**2.02**	1.91	1.72	1.39	1.62
#4000	(−2.74)	–	0.24	1.48	1.31	1.26	**1.50**	1.19	0.72	0.96
#4500	(0.50)	–	1.11	1.67	1.92	1.58	**1.98**	1.89	1.83	1.84
#5000	(2.07)	–	0.39	**1.56**	1.09	1.45	1.26	1.19	1.11	1.12
Avg.	(−0.31)	–	0.50	1.46	1.30	1.14	**1.47**	1.27	1.06	1.03
#0092	(1.97)	–	1.34	1.97	2.93	−0.21	2.88	**3.21**	2.26	1.67
#0482	(1.77)	–	1.60	2.31	2.17	**2.42**	2.26	2.34	2.02	2.26
#0548	(0.46)	–	0.93	1.82	1.67	1.74	**1.92**	1.82	1.86	1.82
#1137	(−0.74)	–	0.80	1.39	1.35	1.42	**1.50**	1.42	1.43	1.43
#1145	(3.08)	–	0.53	1.09	1.00	1.08	**1.12**	1.06	1.06	1.07
#1463	(0.73)	–	−0.16	0.60	0.82	0.26	0.95	**1.04**	0.74	0.92
#2239	(1.52)	–	0.71	1.34	1.28	1.32	1.53	**1.55**	1.50	1.51
#2912	(0.85)	–	1.36	2.24	2.08	2.21	**2.34**	2.27	2.31	2.31
#3394	(1.38)	–	0.80	1.87	1.73	1.80	**1.97**	1.88	−0.16	1.36
#3419	(0.86)	–	0.83	1.69	1.66	0.40	**1.85**	1.71	1.18	1.62
Avg.	(1.19)	–	0.87	1.63	1.67	1.24	**1.83**	**1.83**	1.42	1.60
m01	(0.67)	–	1.29	**2.13**	1.99	**2.13**	1.98	2.04	1.69	1.95
m02	(0.45)	–	1.71	**2.33**	2.12	2.27	2.21	2.17	2.10	2.11
m03	(−0.55)	–	0.83	**2.03**	1.70	0.21	1.92	1.90	0.33	0.51
m04	(−0.03)	–	1.02	**1.94**	1.70	−0.87	1.92	1.79	1.77	−1.03
m05	(0.95)	–	0.33	**1.24**	0.77	0.74	1.06	0.88	0.96	0.46
m06	(1.85)	–	0.75	**1.97**	1.52	0.11	1.55	1.71	1.42	0.33
m07	(2.72)	–	1.08	**1.97**	1.61	1.85	1.85	1.75	1.77	1.68
m08	(1.83)	–	1.66	**2.55**	2.25	2.58	2.25	2.46	2.15	2.40
m09	(2.70)	–	0.28	**1.89**	1.45	0.52	1.54	1.81	1.29	0.65
m10	(1.05)	–	1.37	**2.61**	2.29	2.64	2.46	2.45	2.42	2.46
Avg.	(1.16)	–	1.03	**2.07**	1.74	1.22	1.87	1.90	1.59	1.15

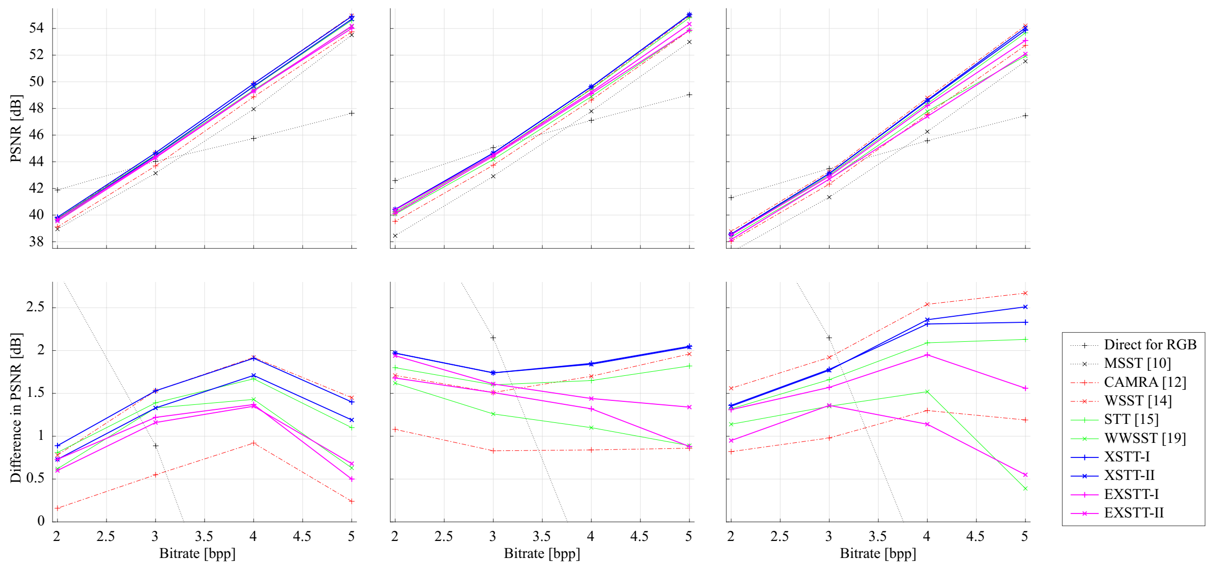

Fig. 12. R-D curves of average bitrate and PSNR in lossy compression: (top) bitrate vs PSNR, (bottom) bitrate vs difference in PSNR compared with MSST (i.e., assuming that the PSNRs of the MSST [10] are 0 dB), (left-to-right) general images, images with the many edges, and mobile phone images.

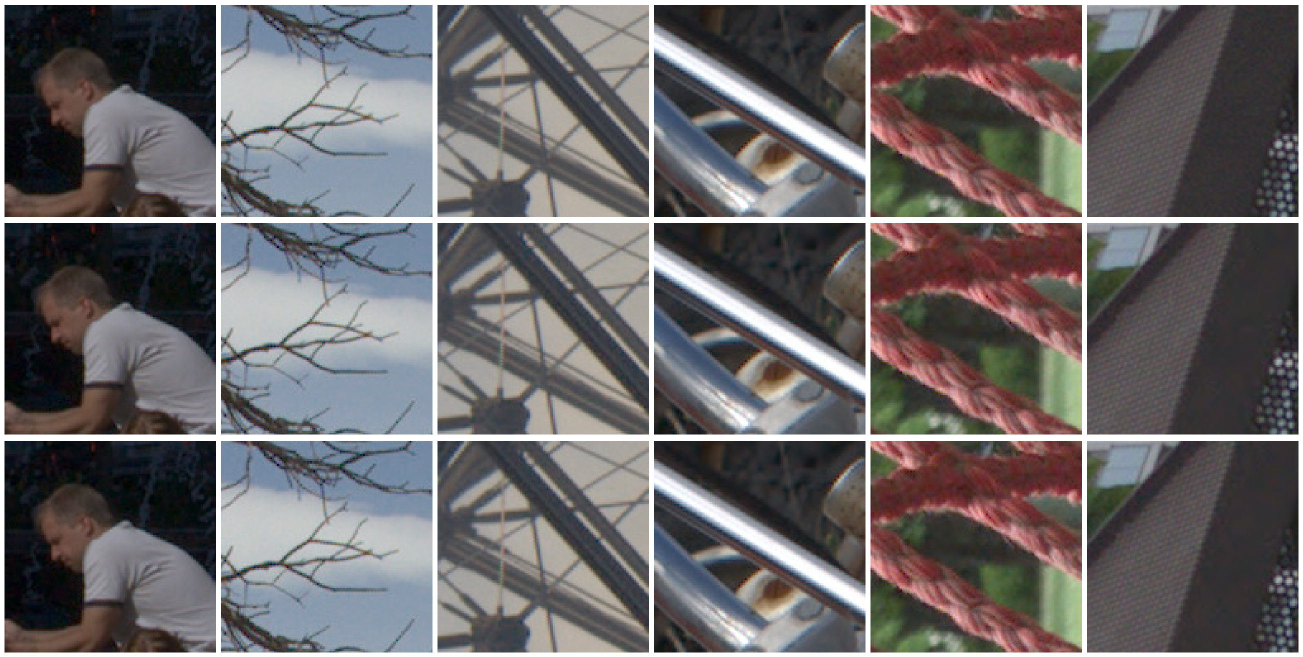

Fig. 13. Part of images developed in lossy compression with 2 bpp: (top-to-bottom) original, XSTT-I, and XSTT-II and (left-to-right) *#2000*, *#4500*, *#0482*, *#2239*, *m04*, and *m09*.