Lossless Compression of Color Filter Array Mosaic Images With Visualization via JPEG 2000

Lossless Compression of Color Filter Array Mosaic Images with Visualization via JPEG 2000

Miguel Hernández-Cabronero*, *Member, IEEE*, Michael W. Marcellin, *Fellow, IEEE*, Ian Blanes, *Member, IEEE*, and Joan Serra-Sagristà, *Senior Member, IEEE*

***Abstract*—Digital cameras have become ubiquitous for amateur and professional applications. The raw images captured by digital sensors typically take the form of color filter array (CFA) mosaic images, which must be "developed" (via digital signal processing) before they can be viewed. Photographers and scientists often repeat the "development process" using different parameters to obtain images suitable for different purposes. Since the development process is generally not invertible, it is commonly desirable to store the raw (or undeveloped) mosaic images indefinitely. Uncompressed mosaic image file sizes can be more than 30 times larger than those of developed images stored in JPEG format. Data compression is thus of interest. Several compression methods for mosaic images have been proposed in the literature. However, they all require a custom decompressor followed by development-specific software to generate a displayable image. In this paper, a novel compression pipeline is proposed that removes these requirements. Specifically, mosaic images can be losslessly recovered from the resulting compressed files, and, more significantly, images can be directly viewed (decompressed *and* developed) using only a JPEG 2000 compliant image viewer. Experiments reveal that the proposed pipeline attains excellent visual quality, while providing compression performance competitive to that of state-of-the-art compression algorithms for mosaic images.**

***Index Terms*—Image Compression, Color Filter Arrays, Bayer CFA, JPEG 2000**

I. Introduction

Digital cameras typically employ monochromatic light sensors laid out in a 2D array. A color filter array (CFA), most commonly the Bayer CFA [1], is situated between the lens and the array of sensors, as depicted in Fig. 1. The CFA filters the incident light so that each sensor element is excited only by photons of one primary color (normally red, green or blue). The resulting single component image can be thought of as a 2D mosaic of red, green and blue pixels, and is thus referred to as a *mosaic image*. Note that in Fig. 1, the green pixels are divided into two subsets labeled G and g. This is done solely for the convenience of discussions in subsequent sections.

Like traditional film negatives, mosaic images must undergo several transformations before they can be viewed. By analogy, these transformations are hereinafter referred to as the *digital development* process. Several considerations crucial to the aspect of the final image are made during digital development. These include color, brightness and sharpness adjustments. Photographers often test different configurations as a part of their creative process. Similarly, scientists working with digital cameras attached to microscopes sometimes need to modify the development parameters to adequately display objects of interest. Additionally, the algorithms on which the development process is based are being actively researched [2]–[14], so that future development processes may yield developed images with higher visual quality, sharper edges and less noticeable chromatic artifacts. For these and other reasons, it is desirable to have the ability to perform future redevelopment. Unfortunately, the development process is not generally invertible. Hence, in many scenarios, it is of paramount importance to store the original mosaic images without loss.

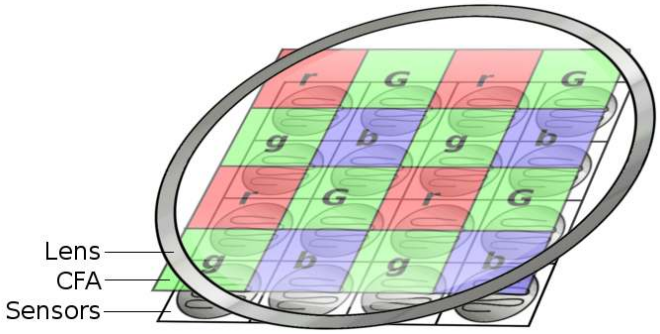


Fig. 1: Diagram of a Bayer CFA over an array of sensors.

Virtually all modern mid-range and high-end digital cameras, including a growing number of mobile phone cameras, allow the user to store the mosaic images in a *RAW format* for subsequent off-camera storage and processing. One main drawback of all existing RAW formats is the need for development-specific software to display the captured images. This precludes RAW (mosaic) images from being easily shared by e-mail or posted on web pages such as on-line photo albums and social networks.

Compression of mosaic images is desirable because it allows more images to occupy a given amount of storage and enables faster transmission, especially over slow channels such as many mobile networks. Most RAW formats feature some type of lossless compression for the mosaic images.

This work has been partially funded by FEDER, the Spanish Government (MINECO) and the Catalan Government under projects TIN2015-71126-R, TIN2012-38102-C03-03, FPU AP2010-0172 and 2014SGR-691.

*M. Hernández-Cabronero, I. Blanes and J. Serra-Sagristà are with the Universitat Autònoma de Barcelona, Bellaterra 08193, Spain (e-mail: mhernandez@deic.uab.cat).

M. W. Marcellin is with the University of Arizona, Tucson, AZ 85721-0104, USA.

For instance, the Canon, Kodak, Nikon and Adobe digital negative (DNG [15]) formats[1] employ non-adaptive Huffman-based compression or directly apply the lossless mode of the JPEG [17] standard (not to be confused with the JPEG-LS [18] or JPEG 2000 [19] standards). These approaches offer fast implementations, but at the cost of reduced compression ratios compared to state-of-the-art algorithms for mosaic images from the literature. Such algorithms include those based on subband coding [20], [21], JPEG [22]–[25], JPEG-LS [26], [27], JPEG-XT [28], JPEG 2000 [26], [29]–[32], vector quantization [33], [34] or Golomb-Rice codes [35], [36].

The best results in the literature are reported by Zhang et al. [21], Bazhyna et al. [27], Chung et al. [36] and Kim et al. [28]. Zhang's method is based on the observation that one level of discrete wavelet transform (DWT) applied to the Bayer CFA mosaic image yields HL_1, LH_1 and HH_1 subbands with approximately the same average energy as the original image. Therefore, additional DWT decomposition levels are applied to all subbands, instead of only to LL_1, as done in a typical dyadic decomposition. Golomb-Rice coding is applied to the resulting wavelet coefficients. On the other hand, [27], [36] and [28] employ approaches based on pixel prediction. In [27], a mean filter is used to interpolate green samples at all non-green positions of the mosaic. The red channel is then predicted using the interpolated green pixels at positions denoted by r in Fig. 1. Similarly, the blue channel is predicted using interpolated green pixels at positions denoted by b in Fig. 1. A Rice code is used for the original green samples (without interpolation) and the prediction errors of the r and b samples. Chung's method is based on the same scheme, but a more sophisticated prediction algorithm is employed. Specifically, neighbors are weighted based on their rank. In turn, the rank of each neighbor is calculated based on how similar that neighbor is to its closest known neighbors. Adaptive Rice codes are then employed for the green samples and the prediction errors. Kim's method relies on the same idea, but employs adaptive arithmetic coding, with 30 context models. These context models are based on edge directionality and the first and second moments of the previous prediction errors.

Although these methods generally yield good compression results, none of them obviate the need for development-specific software for displaying the images, and thus do not enable easy sharing of mosaic images. In this work, a novel lossless compression approach is proposed to address the two main drawbacks of existing RAW formats. Firstly, it allows the display of images directly from compressed files using only a standard JPEG 2000 viewer, without the need for any development-specific software. That is, while specialized techniques are proposed herein for the *encoding* of mosaic images, no such specialized techniques are required for *decoding/viewing* the resulting compressed files. Specifically, standard compliant JPEG 2000 viewers will automatically decompress and carry out an appropriate development process and display a high quality image, directly from the compressed mosaic data. The JPEG 2000 viewer does not require any modification of any kind. Secondly, the original mosaic images can be losslessly recovered from the compressed files, thus preserving the ability to perform full-quality re-development of the original mosaic images. These two goals are achieved from a single compressed file per image, while achieving compression ratios competitive with those of state-of-the-art mosaic-specific image compression algorithms.

[1]A working decoder for each of these formats can be found in the DCRAW [16] software.

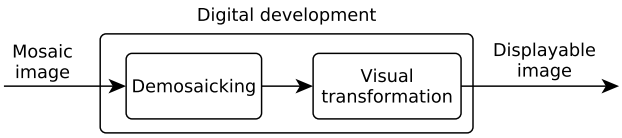


Fig. 2: The digital development process.

Since most modern cameras employ the Bayer CFA [37], this work focuses on only this type of array. Nevertheless, it is straightforward to adapt the proposed method to other CFA types that are based on other color schemes –e.g., cyan, yellow, green and magenta– or that employ different patterns for the mosaic.

The remainder of this paper is structured as follows. The most relevant aspects of the digital development process are detailed in Section II, while the proposed method is described in Section III and its compression performance is evaluated in Section IV. Finally, some conclusions are drawn in Section V.

II. The Digital Development Process

All mosaic images need to be developed before they can be viewed. This development process comprises two consecutive stages, *demosaicking* and *visual transformation*. In the demosaicking stage, the single-component mosaic image is split into separate red, green and blue components. In the visual transformation stage, these components are processed to obtain the final displayable image. A diagram of the digital development process is provided in Fig. 2, and its stages are described next.

A. Demosaicking

Consider a (single-component) mosaic image having $2N \times 2M$ pixels, with each pixel carrying information about a single primary color. Each such pixel is hereinafter referred to as a *red*, *green* or *blue* pixel. In mosaics produced with a Bayer CFA, there are then $N \times M$ red pixels, which correspond to a horizontal and vertical subsampling factor of 2 and are denoted by *r* in Fig. 1. Similarly, there are $N \times M$ blue pixels, which are denoted by *b* in Fig. 1. Finally, there are two sets of green pixels, each of size $N \times M$, denoted by g and G, respectively. Taken together, the red and blue pixels can be thought of as occupying the white squares on a $2N \times 2M$ chessboard. Similarly, the $2(N \times M)$ green pixels occupy the black squares on the same chessboard. The Bayer array employs a higher density of green pixels, as compared to red and blue pixels, due to the higher sensitivity of the human eye to green light [37].

In the usual demosaicking process, the $N \times M$ red pixels from the mosaic image are interpolated to create a red

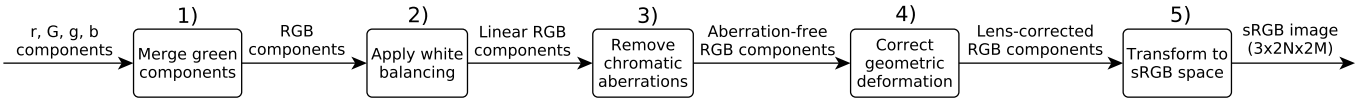


Fig. 3: Visual transformation of r, G, g and b components into a displayable image.

component of size $2N \times 2M$. Specifically, the red pixels are copied from their original positions in the mosaic image to the same locations in the red component. The resulting red component then has pixels only at positions $\{(2i, 2j) : 0 \leq i < N,\ 0 \leq j < M\}$. Similarly, green and blue components of size $2N \times 2M$ are created from the green and blue pixels of the mosaic image, respectively. The $3NM$ missing pixels in each of the red and blue components (as well as the $2NM$ missing pixels in the green component) are then obtained via interpolation.

Several demosaicking algorithms have been recently proposed [4], [6]–[8], [10]–[14] and very detailed reviews of the state of the art can be found in the literature [8], [37]. An alternative approach to demosaicking was described in [23]. In this approach, four components are created. Pixels denoted as r, G, g and b are used to create the first, second, third and fourth component, respectively. An example of this alternative demosaicking with $M = N = 2$ is given by

$$\begin{vmatrix} r_{0,0} & G_{0,0} & r_{0,1} & G_{0,1} \\ g_{0,0} & b_{0,0} & g_{0,1} & b_{0,1} \\ r_{1,0} & G_{1,0} & r_{1,1} & G_{1,1} \\ g_{1,0} & b_{1,0} & g_{1,1} & b_{1,1} \end{vmatrix} \rightarrow \begin{vmatrix} r_{0,0} & r_{0,1} \\ r_{1,0} & r_{1,1} \end{vmatrix} \begin{vmatrix} G_{0,0} & G_{0,1} \\ G_{1,0} & G_{1,1} \end{vmatrix} \begin{vmatrix} g_{0,0} & g_{0,1} \\ g_{1,0} & g_{1,1} \end{vmatrix} \begin{vmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{vmatrix}. \quad (1)$$

Interpolation is then applied to each component separately to yield four components of size $2N \times 2M$.

It is worth noting that, prior to interpolation, the original mosaic image can be recovered from the output components resulting from either approach. However, depending on the interpolation algorithm employed, the original pixel values may not be recoverable after interpolation. Thus, it is of interest to consider compression of the components prior to interpolation.

One advantage of the alternative method from [23] is that the four components are of homogeneous size and have no "missing values." This usually results in superior compression efficiency for compressors such as JPEG 2000 [21], [23]. That is, the four components are typically more compressible than either the three components from the usual demosaicking approach, or the (single component) original mosaic image. Consequently, the alternative demosaicking approach from [23] is discussed exclusively hereinafter.

B. Visual Transformation

Consistent with Fig. 2, the r, G, g and b components that result from the demosaicking process are subjected to visual transformation to produce the final developed image. The typical steps of such a process are depicted in Fig. 3 and described as follows:

1) Green Merging: The two green components are merged into one component –usually by taking their arithmetic mean [16]– without modifying the red or blue components.

2) White Balancing: The components are modified to ensure that gray tones are not perceived with any color tint in the developed image. To this end, the red, green and blue components are multiplied by camera-dependent constants k_r, k_g and k_b, respectively. Usually, these constants are normalized so that $k_g = 1$.

3) Chromatic Aberration Removal: The outputs of the white balancing stage are the *linear RGB* components. They typically present *color fringes* near contrasted edges [5], [9] due to color misalignment [23]. In this stage, these color artifacts are corrected. A review of the most important algorithms for chromatic-aberration removal can be found in [9].

4) Geometric Correction: Depending on the lens attached to the digital camera, a spatial distortion may be present. Even though the distortion is typically small, a geometric correction of the image may be employed to correct it.

5) sRGB Transformation: The linear RGB components that result from the steps above are transformed into a standard color space. To do this, the three pixels (one from each of the three color components) at each spatial position are considered to be a vector that is multiplied by a 3×3 matrix dependent on the camera model and the target color space. Finally, a gamma correction [19] is applied to adjust for any assumed nonlinearity in the brightness of the target display device. Often, the target color space is sRGB [38].

Some variations in the order of the steps can be introduced in the visual transformation process without great changes in the final image. For example, some authors employ a pipeline in which interpolation occurs after white balancing [10], while some software tools merge the green components after removing the chromatic aberrations [16].

III. Compression for Visualization and Re-development

This section proposes a novel compression paradigm for CFA images. In the proposed system, the development process is performed as an integral part of the decompression process of a *standard-compliant viewer*, without requiring any change in the viewer. We analyzed several state-of-the-art standards for this purpose –including JPEG, JPEG-LS, JPEG-XT, HEVC and JPEG 2000– and concluded that only the latter offers enough flexibility to facilitate the display of a high quality *developed* image from a compressed mosaic image file. Therefore, JPEG 2000 is considered exclusively hereinafter.

The development process performed by such a viewer is constrained by the structure of the JPEG 2000 standard. For this reason, the development process is fixed and its parameters cannot be changed at viewing time. As a result, in some cases, the quality of the rendered image may not be as good as that obtained from a specialized digital development process. Nevertheless, the quality obtained is typically high.

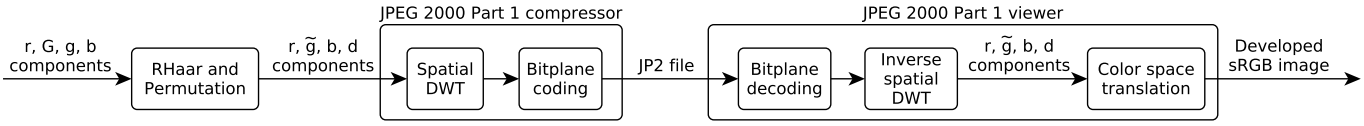


Fig. 4: Main stages of the proposed system using JPEG 2000 Part 1.

Furthermore, an *aware (or smart) decoder* can recover the original mosaic data losslessly. A full-quality development with arbitrary algorithms and parameters can then be performed if desired.

It is worth stressing that, to the best of our knowledge, no system that allows digital development as part of a standard decompression process *and* lossless recovery of the original mosaic data has been previously described, and that only JPEG 2000 offers enough flexibility to meet these goals.

A. Typical JPEG 2000 Compression/Decompression Pipeline

In order to understand the proposed system, it is useful to describe first a typical JPEG 2000 compression and decompression pipeline. Some variations of the proposed method employ a pipeline using technology from only JPEG 2000 Part 1, while other variations add the multicomponent transform extensions from Part 2 of the standard [39]. The first step in the JPEG 2000 pipeline is an optional point transform. When such transform is employed, all components must be of the same size so that a pixel can be defined as a vector containing one sample from each component (all from the same spatial location). The point transform is then applied to each pixel independently. For Part 1 compliance, only the reversible color transform (RCT) or the irreversible color transform (ICT) is allowed [19]. When Part 2 extensions are employed, a rich collection of multi-component transforms (MCT) are available. If desired, multiple MCTs can be applied sequentially. In this case, each such transform is called an *MCT stage*. The components that result from any point transform are then each subjected to the usual 2D JPEG 2000 compression process. For this reason, these components are referred to as the *codestream components*. The 2D JPEG 2000 encoder that is applied to each of the codestream components consists of a *spatial* DWT followed by a bit-plane arithmetic coding process [19].

When a JPEG 2000 decoder is applied to the resulting codestream, the previous steps are inverted in reverse order. That is, inverse bit-plane coding is followed by the inverse *spatial* DWT to obtain the codestream components. Then any MCT, RCT or ICT is inverted as appropriate. It is worth noting that the MCT can only be inverted by Part 2 compliant decoders.

Both JPEG 2000 Part 1 and Part 2 compressed files may include additional information –e.g., ICC color profiles [40], component subsampling factors or component registration offsets– which instruct compliant image viewers as to how components should be displayed.

B. The Proposed System

The JPEG 2000 encoding and decoding pipeline can be exploited so that a standard JPEG 2000 *Part 1* viewer automatically carries out a development process similar to that described in Section II. As stated previously, it is desirable to allow for the lossless recovery of the original mosaic data. Thus, in what follows only reversible (integer-to-integer) transforms are considered in the JPEG 2000 pipeline.

The main stages of the proposed system are depicted in Fig. 4 and described in Subsections III-B1 through III-B3, and summarized in Subsection III-B4 below. Visual results produced by this system are discussed in Subsection III-B5.

1) Demosaicking and Green Channel Merging: Given an original mosaic image, the proposed encoding process begins by extracting the r, G, g and b components according to the alternative demosaicking approach from [23], as described in Section II-A. Then the reversible Haar (RHaar) transform is applied to the G and g components to obtain

$$\begin{aligned} \tilde{g} &= G + \lfloor (g-G)/2 \rfloor \approx (G+g)/2 \\ d &= g - G, \end{aligned} \tag{2}$$

leaving the r and b components unmodified. The four components are then permuted to yield r, $\tilde{g}$, b, d. Clearly, this reversible transform provides some decorrelation between the G and g components that could increase compression efficiency. More importantly, it accomplishes the *Green Merging* stage described in Section II-B. That is, the output $\tilde{g}$ of the reversible transform is (within rounding) the arithmetic mean of the G and g components.

2) JPEG 2000 Part 1 Compression: The r, $\tilde{g}$, b, d components resulting from the previous step are then subjected to compression with a standard lossless JPEG 2000 Part 1 compressor without any MCT or color transform. Note that using a lossless compression algorithm allows perfect recovery of the original mosaic image when desired.

When a standard JPEG 2000 Part 1 viewer is used to display the compressed image, the bit-plane coding and spatial DWT are inverted to recover the codestream components r, $\tilde{g}$, b and d. At this point in the pipeline, an aware (non-standard) decoder could easily recover the original r, G, g and b components and re-create the original mosaic image (and/or perform a sophisticated development process with any desired parameter selections). On the other hand, a standard (unaware) viewer would (automatically) ignore the fourth component d, and continue the rendering process using only the first three components r, $\tilde{g}$ and b.

3) Color Space Translation: As explained in Section II-B, after the green merging stage, the data undergo a white balancing process followed by corrections for aberrations and geometry. The resulting linear RGB data are then remapped to a standard color space such as sRGB. Neglecting aberration and geometry corrections, a standard JPEG 2000 Part 1 viewer can be made to carry out these steps (automatically) by including carefully crafted color space information in the

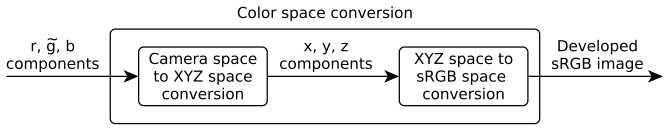


Fig. 5: Color space conversion in JPEG 2000.

compressed file. This can be accomplished by inserting an ICC color profile in the compressed file [19].

An ICC profile contains the necessary information to allow a mapping of the first three codestream components to the (linear) CIE XYZ color space [40]. A compliant viewer uses this information to map the first three codestream components to an appropriate color space for the target display. Conceptually, this can be accomplished by first mapping to linear XYZ space, and then to the target space, as shown in Fig. 5. A particular type of ICC profile supported by JPEG 2000 includes a 3×3 matrix I together with three non-linearities. The information in the profile is used to map the three image components to linear XYZ space as follows: The first nonlinearity is applied to the first color channel, the second nonlinearity to the second color channel, and the third nonlinearity to the third color channel. Each resulting pixel is then transformed via the 3×3 matrix. It is worth noting that the design of the required nonlinearities, as well as the matrix I, is specific to the camera used to acquire the original mosaic image (but not to individual mosaic images from that camera). Any white balancing required for a given camera can be included in the matrix I. After bringing the image data to linear XYZ, the image viewer then applies whatever color space conversion is necessary to bring the image data to a color space that is appropriate for the target display. A reasonable choice for many computer monitors is the sRGB color space, which requires multiplication by an additional 3×3 matrix followed by gamma correction [19].

4) Summary: To recap, the proposed *encoder* splits the mosaic image into four image components r, G, g and b. A reversible Haar transform is applied to G and g to yield $\tilde{g}$ and d. The components are then ordered as r, $\tilde{g}$, b, d prior to compression with JPEG2000. An ICC profile, specific to the camera used to acquire the mosaic image, is embedded in the resulting JPEG 2000 file. An *unmodified* standard JPEG 2000 Part 1 viewer can then be used to render an image (without the use of any other tools such as an inverse Haar transform). An aware (smart) decoder can employ an inverse Haar transform to obtain lossless decompression of the original mosaic image.

5) Visual Results: Several images compressed with the proposed encoder and then rendered using the JPEG 2000 compliant application `kdu_render` [41] are shown in Fig. 6. It can be observed that the images are perceptually pleasing, with correct white balance and no annoying visible artifacts. In the interest of space, the images of this figure are displayed at reduced size (spatially downsampled). Fig. 7 depicts a crop from another example image at full-size, with no spatial downscaling. Part (a) of the figure shows the results of the proposed pipeline as described above. For the purpose of comparison, part (b) shows the results of the popular development-specific software DCRAW [16]. The latter has been invoked so that interpolation is skipped in the demosaicking step and the image is output in the standard sRGB color space instead of the BT.709 color space, employed by default.[2] Both `kdu_render` and DCRAW produce 8-bit color components. As is obvious from the Figure, it is difficult for the naked eye to find any difference between the two image versions. These results suggest that the proposed system is able to successfully complete a digital development process and produce visually satisfactory images.

C. Interpolation Within the Pipeline

The proposed pipeline as discussed above does not include interpolation. Thus, the resulting developed images have size $N \times M$, whereas the default behavior of DCRAW and other development software tools includes interpolation so that the resulting developed image is of size $2N \times 2M$. Nevertheless, it is also possible to include interpolation as an integral part of the standard JPEG 2000 viewer pipeline so that an image of size $2N \times 2M$ is produced by default.

A straightforward approach, compliant with Part 1 of the standard, is to define horizontal and vertical subsampling factors of 2 for all components in the JPEG 2000 canvas coordinate system [19]. These factors do not affect the compression process, but instruct compatible viewers to put pixels from positions (i, j) of the decompressed image into positions $(2i, 2j)$ of the rendered image, of size $2N \times 2M$. Odd rows and columns are then interpolated by the viewer. This method is hereinafter referred to as *Part 1 interpolation* (without component registration (CRG)). The main drawback of this approach is the fact that the relative positions of the different colors in the Bayer CFA are disregarded. For instance, pixels $r_{0,0}$, $G_{0,0}$, $g_{0,0}$ and $b_{0,0}$ are co-located in the interpolated image although they are registered at different spatial positions on the sensor. A partial solution to this problem, also Part 1 compliant [19], is to make use of the CRG feature of JPEG 2000. With this feature, viewers can be instructed to apply integer vertical and horizontal offsets individually to each color component. In what follows, this is referred to as Part 1 interpolation *with CRG*. Unfortunately, this solution is not entirely satisfactory. The $\tilde{g}$ samples must be assigned a single fixed offset even though they contain information from both the G and g samples, which come from different spatial locations on the sensor.

Sample image crops reflecting the interpolation discussion above are provided in Fig. 8. Results for *Part 1 interpolation* and *Part 1 interpolation with CRG* are shown for the *D60* image in Figs. 8a and 8b, respectively. Results for a different image captured by a Nikon *D50* camera are shown at 400% magnification in Figs. 8e and 8f. Also included for comparison are the results for the same images from the development-specific DCRAW software. These results are shown in Figs. 8d and 8h, respectively. The reader is encouraged to examine all these images with magnification set to exactly 100% in her/his PDF viewer. This is to avoid the effect of any interpolation or

[2]The `-h` and `-w -W -g 2.45 12.92` parameters are invoked to disable the interpolation and to use the sRGB color space, respectively.

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

(e) (f) (g) (h)

Fig. 6: Images compressed with the proposed encoder without interpolation. The original mosaic images were captured by the Nikon (a) D40, (b) D70S, (c) D100, (d) D200, (e) D300, (f) D3100, (g) D3200, and (h) D5200 camera models.

(a) (b)

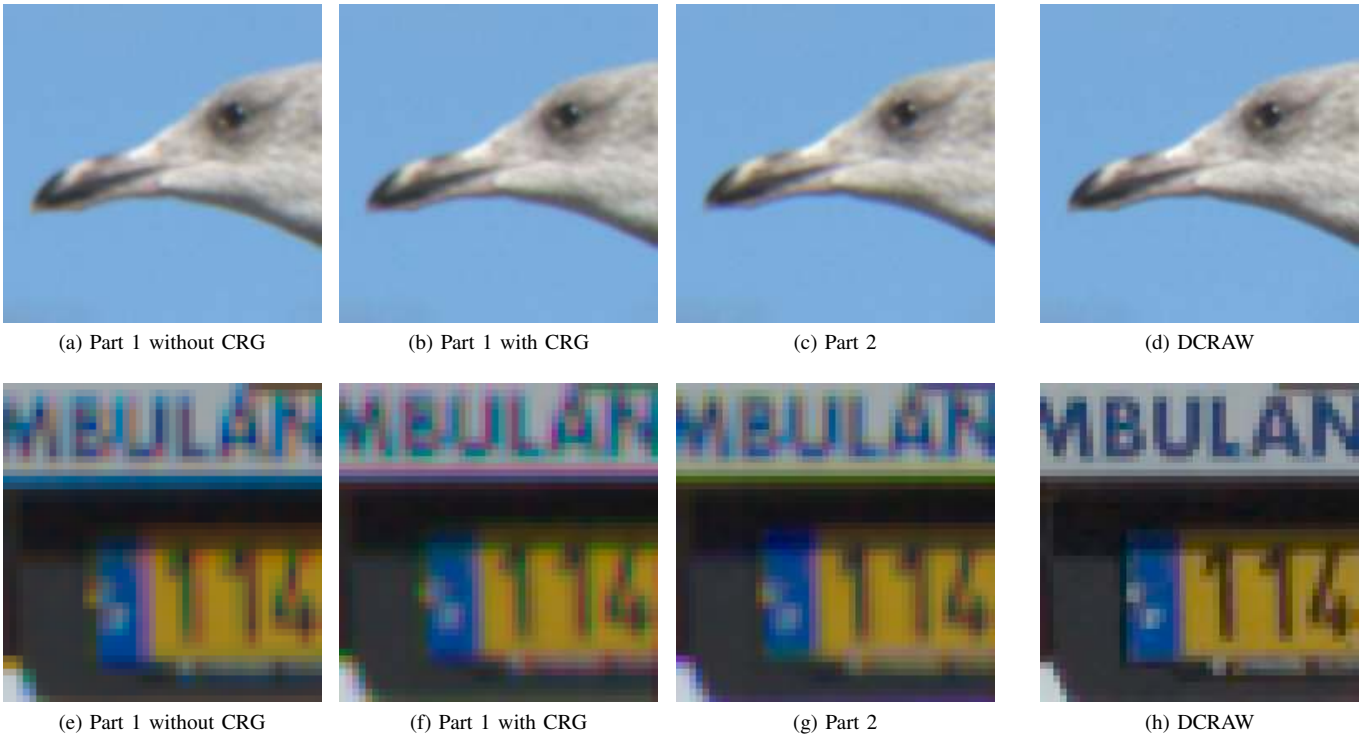
Fig. 7: Crop (385×385) of a Nikon D60 image (a) rendered by the proposed pipeline without interpolation and (b) generated by DCRAW without interpolation.

decimation performed by the PDF viewer. It can be observed that, at original size, both Part 1 interpolation strategies yield visually pleasant results, similar to those generated with DCRAW. It can also be observed that, for the high-contrast regions shown in the magnified image, color fringes and other artifacts are produced by the *Part 1 interpolation (without CRG)* strategy. Careful observation reveals that *Part 1 interpolation with CRG* reduces these artifacts and produces slightly sharper images. Notwithstanding, artifacts are still present, due mainly to green component misalignment.

D. Component Alignment Using JPEG 2000 Part 2

Even though the visual quality produced by the approach described above is sufficient for many usage scenarios, it is possible to use Part 2 of the JPEG 2000 standard to correctly align all color components and increase the quality of the developed images. In this interpolation strategy –hereinafter referred to as *Part 2 interpolation*– the G and g components are upsampled and aligned *before* merging them, thus removing the aforementioned component misalignment.

Some changes in the proposed pipeline are necessary to at-



(a) Part 1 without CRG (b) Part 1 with CRG (c) Part 2 (d) DCRAW

(e) Part 1 without CRG (f) Part 1 with CRG (g) Part 2 (h) DCRAW

Fig. 8: Comparison of interpolation strategies for a Nikon D60 image (top row, original size) and a Nikon D50 image (bottom row, 400% magnification).

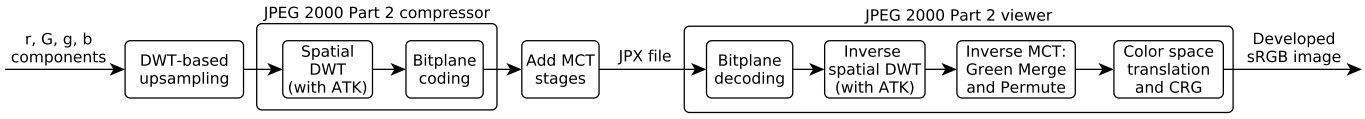


Fig. 9: Pipeline with correct component alignment using JPEG 2000 Part 2.

tain correct component alignment. A diagram of the enhanced pipeline is depicted in Fig. 9. A significant difference in this pipeline, compared to the Part 1 compliant pipeline, is that green merging is not performed by the encoder. Rather, green merging is performed by the image viewer using an inverse multicomponent transform (MCT). Additionally, interpolation is not performed by the viewer. Rather, it is performed by the encoder by means of an inverse 5/3 DWT. Specifically, each of the four $N \times M$ components extracted from the mosaic image is upsampled by considering it to be the LL subband of a discrete wavelet decomposition. Three additional subbands LH, HL and HH are created, each having all coefficients equal to zero. One level of inverse 5/3 DWT is then performed to obtain an interpolated component of size $2N \times 2M$. In order to properly align component g, a novel modification of the usual 5/3 DWT is proposed. A full description of this transform is provided in the appendix. As before, r and b can be aligned via CRG offsets. All four interpolated components are then subjected to compression using a standard JPEG 2000 Part 2 encoder, with no point transform (RCT or MCT) employed. By construction, when the (appropriate) spatial 5/3 DWT is applied to the r, G, g and b components, the subbands of the first decomposition level (e.g., LH_1, HL_1 and HH_1) are all identically zero. The JPEG 2000 bit-plane encoder is able to encode these all zero subbands very efficiently. For this reason, the resulting file size is virtually unchanged by the inclusion of interpolation. Specifically, the bit-rate required for lossless compression is increased by only about 0.001 bits per pixel per component.

As mentioned previously, no MCT is performed during encoding. However, consistent with Fig. 9, the headers of the compressed file are modified to include an MCT stage which instructs a JPEG 2000 Part 2 viewer to perform an inverse MCT that acts to merge the green components. Specifically, when the resulting file is opened with a JPEG 2000 Part 2 viewer, the bitplane coding and the spatial DWT are inverted, followed by the inverse MCT. The MCT is constructed so that it performs the RHaar transform and the permutation described in Section III-B. White balancing and transformation to the sRGB color space are applied via ICC color profile. A standard compliant JPEG 2000 Part 2 viewer will automatically decompress and display an image of size $2N \times 2M$, with all color components appropriately registered. An aware decoder can recover the original mosaic data losslessly by performing one less than the number of spatial inverse wavelet transform levels indicated by the code stream header, and omitting the

TABLE I: Average lossless compression results in bpppc (lower is better) for real Bayer CFA mosaic images.

Vendor	# Images	Proposed encoder		RKLT + JPEG 2000 [19]	JPEG-LS [18]	JBIG2 [42]	HEVC [43]	JPEG-XT [44]
		Part 1	Part 2					
Canon	20	7.295	7.319	7.260	7.120	7.497	8.105	7.847
Fuji	20	8.718	8.725	8.572	8.530	8.663	10.681	9.291
Nikon	20	7.264	7.309	7.205	7.187	7.627	8.328	7.851
Olympus	20	6.482	6.516	6.353	6.362	6.793	7.006	7.030
Sony	20	6.169	6.176	6.099	6.094	6.380	6.522	6.737
All vendors	100	7.045	7.068	6.963	6.927	7.264	7.905	7.612

green merging transform.

Crops of the *D60* and *D50* images, developed with the proposed Part 2 interpolation strategy, are shown in Figs. 8c and 8g, respectively. It can be observed that the *D60* image crop shown at original size is very similar to those rendered using the DCRAW or Part 1 interpolation strategies. Even though some differences are evident among the *D60* images, when compared closely, side by side, all images are generally of high quality and are free from visually annoying artifacts. Based on observations of these crops and of further images provided as supplementary materials, it is reasonable to conclude that the proposed pipeline yields visually pleasing images under normal viewing conditions. In order to see the improvements of Part 2 interpolation over Part 1 strategies, magnified high-contrast regions are shown in Fig. 8g. As can be seen there, Part 2 interpolation reduces the color aberrations and jagged edges produced by the Part 1 interpolation strategies. These improvements result in clearer details and a sharper overall look, as is readily apparent upon examination of the letters "BU" that appear as text along the top of the images in Figs. 8e and 8f. The remaining visual differences between the Part 2 approach and DCRAW (Fig. 8h) are due to the more sophisticated interpolation algorithm employed by the latter [2], which removes chromatic aberrations (due to the camera) and produces sharper edges. To the best of our knowledge, existing chromatic aberration techniques are not reversible [4]–[10], and thus, not admitted by the framework of the JPEG 2000 standard. Despite the fact that not all chromatic aberrations are removed by the proposed system, we emphasize again that the system provides automatic developing and rendering of mosaic images using only a standard viewer. If higher quality rendering is desired, an aware decoder can always decompress the original Bayer data losslessly and render them via any current or future digital development process.

IV. Compression Performance

A. Lossless Compression Performance

Many algorithms have been developed for mosaic images. While these algorithms are suitable for use on real mosaic images, in many cases, results have been reported only for simulated mosaic images. These simulated mosaic images were obtained by downsampling fully developed 8-bit sRGB images [20]–[22], [26]–[28], [35], [36], [45]. This approach is reasonable, but not ideal since the development process described in Section II cannot be inverted perfectly [10], [30]. Thus, simulated mosaic images may contain statistically significant differences from real mosaic images.

In our work, results are presented for the common simulated mosaic images for the purpose of comparison with results from the literature. Additionally, we report results for 100 real Bayer CFA images.[3] Each real mosaic image was produced with a different camera model by Canon, Fuji, Nikon, Olympus or Sony. Image dimensions range from 1440×1064 to 6036×4020, and samples are stored using 10 bps, 12 bps or 14 bps. The images have been selected to depict a variety of scene types, illumination, and color and edge complexity. Thumbnails for 8 of these images were shown previously in Fig. 6. Compressed files obtained by applying the proposed encoders to each of these CFA images can be directly displayed using the Kakadu `kdu_show` tool, or decompressed directly to 8 bps sRGB image files using the `kdu_render` tool [41]. Sample original images, their corresponding compressed files (for both the Part 1 and Part 2 versions of the proposed encoder) and all implementation details (including the ICC profiles, etc.) are available as supplementary content at http://ieeexplore.ieee.org.

1) Real Mosaic Images: Lossless compression results for the proposed encoder (using each of the two proposed interpolation strategies) are provided in Table I. The results presented in the table include all overhead due to the definition of the required ICC color profiles, CRG offsets and MCT stages. The bit rate for a given image is calculated as the size of the compressed file in bits, divided by the number of pixels in the original mosaic image. This quantity has units of bits per pixel (bpp). The same numerical result would be obtained by dividing the file size in bits by the number of pixels in each of the four components extracted from the mosaic, with the resulting quantity then divided by 4. In light of this, the bit rate can also be interpreted as having units of bits per pixel per component (bpppc) with respect to the four-component image. We emphasize that we avoid defining the bit rate according to the size of the developed image, as displayed by the viewer, as this size differs depending on the method employed. Average bit rates are reported over certain collections of images in the table. These averages are computed as the sum of the file sizes (in bits) of the compressed files divided by the sum of the pixel counts of the corresponding mosaic images. This is in contrast to averaging the bit rates from the individual images.

For comparison, results for several other image compression algorithms are also reported in Table I. The input to each algorithm is the four component image extracted from the mosaic without interpolation. While other demosaicking

[3]Downloaded from https://rawsamples.ch/index.php/en/

TABLE II: Average lossless compression results in bpppc for (8-bit) developed images as produced by DCRAW, and for the corresponding mosaic images.

Algorithm	DCRAW	Mosaic
RKLT + JPEG 2000 [19]	8.536	6.963
JPEG-LS [18]	8.673	6.927
JBIG2 [42]	9.892	7.264
HEVC (H.265) [43]	10.256	7.905
JPEG-XT [44]	9.734	7.612
Proposed (Part 1)	–	7.045
Proposed (Part 2)	–	7.068

TABLE III: Average lossless compression results in bpppc for 6 simulated mosaic images. The *Part 1* and *Part 2* columns correspond to the proposed encoders.

Image	[21]	[36]	[27]	[28]	Part 1	Part 2
Boat	5.028	4.881	4.984	4.793	5.170	5.424
Fence	4.823	4.711	4.886	4.649	4.663	5.238
Landscape	6.243	6.138	6.279	6.072	6.047	6.817
Lighthouse	4.867	4.803	4.864	4.699	4.181	5.268
Wall	5.650	5.478	5.750	5.438	6.044	6.205
Windows	5.725	5.570	4.984	5.506	6.270	6.431
Average	5.389	5.264	5.291	5.193	5.396	5.897

strategies may be used, for consistency, the four-component demosaicked images are used throughout this paper.

Algorithms employed in the comparison include JPEG-LS [18], JBIG2 [42], HEVC (H.265) [43], JPEG-XT [44] and JPEG 2000 using the four component reversible Karhunen-Loève Transform (RKLT). As discussed previously, the results for ordinary JPEG 2000 Part 1 with no interpolation and no exploitation of intercomponent correlation are essentially the same as those reported for the proposed Part 2 encoder, and so are not reported separately.

As can be seen in Table I, the best lossless compression performance is produced by JPEG-LS followed closely by the JPEG 2000 schemes. Averaged over all images, JPEG-LS and JPEG 2000 with the RKLT yield essentially identical performance. Specifically, they are within 0.04 bpppc or 0.52% of each other. The proposed Part 1 and Part 2 encoders introduce, respectively, a loss of 0.082 bpppc (1.2%) and 0.105 bpppc (1.5%) as compared to JPEG 2000 with the RKLT. This suggests that the proposed encoders provide self-developing capabilities with a negligible penalty in compression performance.

Recall that the proposed Part 1 encoder includes a reversible Haar transform which serves to merge the two green components. As a side benefit, a small compression gain is obtained from the resulting exploitation of intercomponent correlation between the green components.[4]

The results for JPEG 2000 with the RKLT were included in Table I primarily to give an indication of further gains that might be achieved by employing point transforms to exploit intercomponent dependence. As discussed above, these improvements are insignificant. Improvements obtained when using the RKLT with other algorithms listed in the table are similar, with one notable exception. For HEVC, use of the RKLT results in a significant deterioration in compression performance. Additional point transforms (including the RHaar transform, the RCT and the 5/3 DWT) were tested and found to yield even smaller improvements. Indeed, the RCT actually causes a slight deterioration in compression performance for all algorithms tested.

2) Developed Images: As discussed in Section I, mosaic images are traditionally developed with the resulting images then subjected to compression. For the purpose of comparison with the proposed pipeline, all 100 mosaic images were developed with DCRAW. The resulting images were stored using 8 bits per sample and then compressed with JPEG 2000, JPEG-LS, JBIG2, HEVC and JPEG XT. Average results for each algorithm are provided in Table II. Average results for the corresponding mosaic images are repeated from Table I for ease of comparison.[5] As is obvious from the table, compressing mosaic images prior to development produces superior results compared to compressing developed images. This result holds despite the fact that the mosaic images have a bit depth of 12 to 14, while the developed images have a bit depth of only 8. This behavior can be attributed to redundancy introduced by the interpolation performed during the development process, which is not completely removed during compression. This is consistent with previous work [20]–[27], [29]–[36], [45].

3) Simulated Mosaic Images: We conclude this section with compression results for simulated mosaic images to provide a comparison to algorithms from the literature that were developed specifically for compression of CFA mosaic images [21], [27], [28], [36] (is in contrast to the standard image compression schemes employed in the comparisons above). In these four publications, results were provided for 6 common color images (Boat, Fence, Landscape, Lighthouse, Wall and Windows) of size 512×768. These images were downsampled to obtain simulated mosaic images, which were then compressed. Results for applying this procedure with our proposed Part 1 and Part 2 encoders are provided in Table III along with results originally reported in [21], [27], [28], [36]. Interestingly, the results indicate that the Part 2 encoder is about 0.5 bpppc worse than the Part 1 encoder for the *simulated* mosaic images. This is in contrast to the results for *real* mosaic images presented in Table I, where the difference is less than 0.02 bpppc. The larger difference for the simulated mosaic images is due to a high degree of correlation that exists between the green components of the simulated images. This correlation is exploited by the Haar transform that is used to merge the green components in the Part 1 encoder. Other results from Table III indicate that the four mosaic-specific algorithms yield average results about

[4]We emphasize that even though the Haar transform is normally considered a JPEG 2000 Part 2 tool, the proposed system is still Part 1 compliant. This can be seen by noting that the Haar transform is used as a preprocessing step prior to a JPEG 2000 Part 1 encoder, and that no inverse Haar transform is used in the decoder.

[5]As discussed above, bit rates are calculated using the number of pixels in the corresponding mosaic images, rather than the number of pixels in the developed images. This always yields a fair comparison (i.e., smaller bit rates always imply smaller file sizes) by avoiding the issue of different development processes that may yield different final image sizes. This also explains why the resulting bit rates can be larger than 8 bits per sample.

0.01 bpppc, 0.13 bpppc, 0.10 bpppc and 0.20 bpppc better than the proposed Part 1 encoder, respectively. It is possible that these small differences may be bridged by exploiting further correlation that may exist between other components in the simulated images. This is not explored further as such correlations do not seem to exist in real mosaic images.[6]

Everything considered, it can be concluded that the proposed encoders produce competitive lossless compression results as compared to those of both standard and mosaic-specific compressors.

B. Rate-Distortion Performance

As discussed throughout this paper, images compressed by the proposed encoder are fully JPEG 2000 compliant. Hence, it is possible to use any of the scalability features of the standard, including scalability by resolution, spatial region, and quality. This is particularly useful in the context of a JPIP client/server scenario [46] where large images can be viewed remotely using zoom, pan and quality progressivity. In this context, it is interesting to explore the rate-distortion performance of the proposed system under progressive (lossy-to-lossless) transmission. To this end, a given mosaic image was losslessly compressed with the proposed Part 2 encoder. The resulting (single) compressed file was then decompressed/rendered at increasing target bitrates to render a series of 24-bit (eight bit per component) sRGB images of increasing qualities. The quality of these images was then evaluated (with reference to an "original" sRGB image) via the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM [47]). The following definition of the PSNR between an original image I and a reconstructed image $\hat{I}$ was used:

$$\mathrm{PSNR}(I,\hat{I}) = \log_{10}\frac{\max(I)^2}{\mathrm{MSE}(I,\hat{I})}, \tag{3}$$

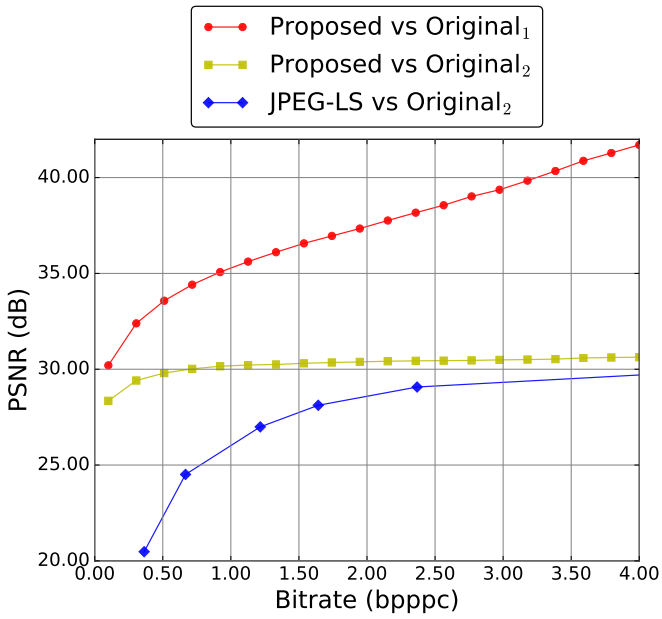
where $\max(I) = 255$ is the maximum possible pixel value of I and $\mathrm{MSE}(I,\hat{I})$ is the mean squared error, defined as

$$\mathrm{MSE}(I,\hat{I}) = \sum_{x,y,z} |I_{x,y,z} - \hat{I}_{x,y,z}|^2. \tag{4}$$

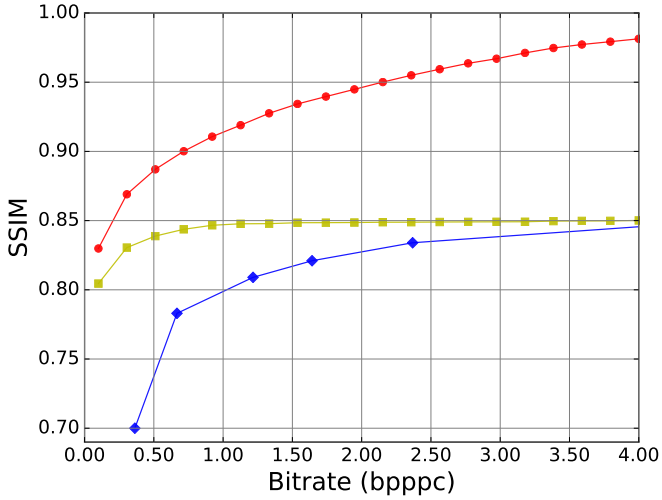
SSIM results were obtained using the Matlab 2016a `ssim` routine. This procedure was repeated for each of the original mosaic images.

The PSNR and SSIM for each target bitrate, averaged over all 100 images, is shown in Fig. 10a and Fig. 10b, respectively, for two different choices of "original" sRGB images. The first such choice is the image obtained from full (lossless) decompression and rendering via the Part 2 pipeline shown in Fig. 9. The second choice for "original" image is obtained by rendering via DCRAW. In what follows, these are referred to as $\mathrm{Original}_1$ and $\mathrm{Original}_2$, respectively. It is worth noting that $\mathrm{Original}_1$ corresponds to the imagery depicted in Figs. 8c and 8g, while $\mathrm{Original}_2$ corresponds to the imagery depicted in Figs. 8d and 8h.

[6]Another anomaly associated with the simulated images is that the Lighthouse image, when compressed with the proposed pipeline, contains significant annoying chromatic artifacts. No such extreme artifacts have been observed over a wide variety of real CFA images.



(a)



(b)

Fig. 10: Average rate-distortion results for the proposed Part 2 encoder. (a) PSNR results; (b) SSIM [47] results.

As expected, the PSNR and the SSIM both increase with bit rate for both choices of original image. More specifically, the PSNR with respect to $\mathrm{Original}_1$ increases rapidly while that for $\mathrm{Original}_2$ reaches a limit of about 32 dB, corresponding to the differences observed between Figs. 8g and 8h. Although not shown in the figure, the PSNR with respect to $\mathrm{Original}_1$ approaches infinity as the bit rate approaches that required for lossless decompression of the original mosaic data. Consistent with this, the SSIM approaches its maximum value of 1 with respect to $\mathrm{Original}_1$ as the target bitrate is increased, while converging to a lower value of about 0.85 with respect to $\mathrm{Original}_2$.

Rate-distortion results for the mosaic-specific methods [21], [27], [28], [36] cannot be provided because they are purely lossless algorithms. Among the algorithms compared in Ta-

TABLE IV: Average execution time in seconds for lossless compression of real Bayer CFA mosaic images.

Vendor	# Images	Proposed encoder		RKLT + JPEG 2000 [19]	JPEG-LS [18]	JBIG2 [42]	HEVC [43]	JPEG-XT [44]
		Part 1	Part 2					
Canon	20	1.32	1.60	1.38	2.00	3.07	57.41	3.56
Fuji	20	1.20	1.52	1.23	1.68	2.76	47.83	3.33
Nikon	20	1.10	2.05	1.58	2.35	3.61	66.98	4.10
Olympus	20	1.20	1.80	1.22	1.77	2.63	51.20	2.95
Sony	20	1.77	2.49	1.92	3.04	4.42	87.83	4.94
All vendors	100	1.32	1.75	1.47	2.17	3.30	62.25	3.78

ble III, JPEG-LS provides the best lossless performance. Results for lossy compression via JPEG-LS are included in Figures 10a and 10b. Since JPEG-LS does not provide any type of scalability, the results in the figures were obtained by compressing and decompressing each image repeatedly employing different values of the near lossless parameter d [18]. As can be observed, JPEG-LS provides lower PSNR and SSIM results than the proposed method at all tested bitrates.

C. Complexity

The complexity of each proposed encoder is essentially the same as that of a standard JPEG 2000 encoder. In the proposed Part 1 encoder, the reversible Haar transform is followed by a standard JPEG 2000 Part 1 encoder. The complexity of this Haar transform is linear ($\mathcal{O}(N)$) in the number of pixels in the image, and is actually lower than the complexity of the RCT which would normally be used to compress color imagery with JPEG 2000, but is omitted by the proposed Part 1 encoder. In the proposed Part 2 scheme, one level of inverse DWT is applied prior to JPEG 2000 compression. This also introduces additional complexity that is only linear in the number of pixels. In the *Add MCT stages*, the headers of the compressed file are modified to include the required *Green Merge and Permute* inverse MCT. The overhead due to this stage does not depend on the number of pixels of the image ($\mathcal{O}(1)$). By construction, the decoders for both schemes are unmodified JPEG 2000 decoders, and thus have no additional complexity.

Table IV provides execution times for all algorithms considered in Table I. Results were obtained on a dedicated Intel Core i7-6600U CPU @ 2.60 GHz machine with 16 GB of RAM. The values in this table must be interpreted with caution. The software implementations used in gathering this information may reflect dramatically different levels of optimization, and thus, may not accurately reflect inherent complexity differences between the algorithms. Nevertheless, the execution times indicate that the proposed algorithms entail reasonable levels of complexity. According to the table, the proposed Part 1 encoder is the fastest algorithm. This is due primarily to the highly optimized software development toolkit (Kakadu [41]) employed in our implementation. We note that the Part 2 encoder includes our own (much less optimized) implementation of the inverse DWT used for upsampling.

Execution times for [21], [27], [28], [36] and for the proposed encoders are provided in Table V. We note here that results for [21], [28] were reported in the original papers as a fraction of the compression time of standard JPEG-LS or JPEG 2000. Since implementations of [21], [28] are not available, we have calculated the values in Table V by measuring execution times for JPEG-LS and JPEG 2000 and multiplying by the reported fractions. To provide results for [27], we have implemented the prediction algorithm described in that publication. Results for [36] have been obtained with the authors' implementation.[7] Results were obtained on the same dedicated machine as above. As before, considerable caution should be used in the interpretation of these execution times. However, it is safe to say that the algorithms reported in [21] and [28] have significantly lower complexity than the proposed system. This can be explained by the low-complexity entropy coders employed by [21], [28] rather than the context-based arithmetic coding in JPEG 2000.

TABLE V: Average execution time in seconds for 6 simulated mosaic images. The *Part 1* and *Part 2* columns correspond to the proposed encoders.

Image	[21]	[36]	[27]	[28]	Part 1	Part 2
Boat	0.061	0.118	0.135	0.032	0.111	0.233
Fence	0.109	0.202	0.215	0.025	0.100	0.220
Landscape	0.046	0.085	0.108	0.030	0.121	0.338
Lighthouse	0.043	0.074	0.103	0.028	0.104	0.374
Wall	0.046	0.079	0.110	0.034	0.109	0.243
Windows	0.054	0.099	0.124	0.038	0.147	0.274
Average	0.059	0.113	0.132	0.034	0.115	0.280

V. Conclusion

Mosaic images captured by digital sensors need to be developed before they can be displayed. The development process is not reversible and, in practice, photographers and scientists often develop images several times using different parameter choices. Hence, it is often desirable to store the original mosaic images losslessly. Since uncompressed mosaic images are more than 30 times larger than developed images stored in JPEG format, data compression is a valuable approach to cope with the storage and transmission of mosaic images. Although several compression algorithms have been proposed in the literature, they require specific decoders and development-specific software tools to visualize the compressed mosaic images. In this paper, a novel compression approach based on JPEG 2000 is proposed. Unlike existing methods, the compressed files produced by this method can be directly displayed using any JPEG 2000 standard-compliant viewer without need for additional development software. In terms of visual

[7] Available at http://www.eie.polyu.edu.hk/~enychan/.

quality, the proposed method is comparable to development-specific software. In terms of lossless compression performance, results are within 0.20 bpppc of the best published algorithms. A useful property of the proposed pipeline is that compressed images can be rendered in a progressive fashion, again using only a standard JPEG 2000 viewer. In summary, the proposed technique is the first to include the development process as an intrinsic part of its compression/decompression pipeline and offers competitive visual quality and compression performance. It is worth reiterating that the original Bayer data are stored losslessly in the proposed codestream format. Accordingly, in addition to the high quality rendering that can be performed by any JPEG 2000 standard decoder, an aware decoder can recover the original Bayer data and perform any desired rendering algorithm.

APPENDIX
MODIFIED DWT FOR INTERPOLATION

The DWT-based interpolation methods employed by the proposed Part 2 pipeline are described in this appendix. As described in Section III-D, components r, G and b are interpolated via the *standard* inverse 5/3 DWT, while the g component is interpolated via a *modified* inverse 5/3 DWT. The main contribution of this appendix is the description of the modified inverse 5/3 DWT. For completeness, and for ease in describing the modified transform, we begin with a description of the standard inverse transform.

Since the DWT as supported by JPEG 2000 is separable, it suffices to describe the lifting network used to implement the 1D inverse transform. Let $y[i]$ be the i-th sample of the interleaved sequence to be transformed. In this sequence, the samples at even positions (i.e., samples having even indices) are low-pass, while the samples at odd positions are high-pass. The standard inverse transform first updates the samples at even positions as [19]:

$$y[2n] \leftarrow y[2n] - \left\lfloor \frac{y[2n-1] + y[2n+1]}{4} + \frac{1}{2} \right\rfloor. \quad (5)$$

The samples at odd positions are then updated as:

$$\begin{aligned} y[2n+1] &\leftarrow y[2n+1] - \left\lfloor -\frac{y[2n] + y[2n+2]}{2} + \frac{1}{2} \right\rfloor \\ &\approx y[2n+1] + \frac{y[2n] + y[2n+2]}{2}. \end{aligned} \quad (6)$$

Recall that the interpolation process described in Section III-D treats data to be interpolated as being low-pass, introduces zeros in place of high pass data, and performs one level of inverse DWT. In light of this (and neglecting rounding), the resulting interpolator is depicted in Fig. 11. In this figure, the values of $x[i]$ at the input (top) of the lifting network represent the data to be interpolated. The values at the output (bottom) represent the interpolated data. Solid lines in the figure indicate multiplication by the adjacent value, while dashed lines indicate that a given sample is not updated in the corresponding lifting step.[8] As can be observed at the output

[8]In Fig. 11, the second multiplier from the left in the first lifting step is $-1/2$ rather than $-1/4$ to account for image boundary conditions [19]. Similarly, in Fig. 12, the second multiplier from the left in the fifth lifting step is 1 rather than $1/2$.

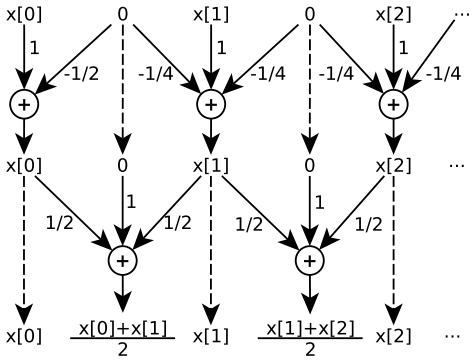


Fig. 11: Interpolation using the *standard* inverse 5/3 DWT.

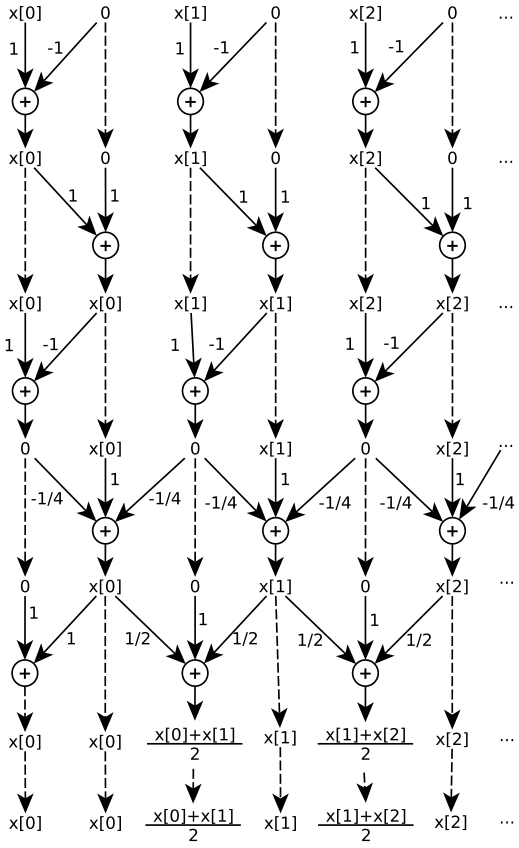


Fig. 12: Interpolation using the *modified* inverse 5/3 DWT.

of the interpolator, the original samples are placed in even positions while odd positions are interpolated as the average of the two nearest neighbors. When extended to two dimensions in a separable fashion, the resulting interpolator places the original samples at positions $(2i, 2j)$, with interpolated values at all other positions.

As also described in Section III-D, a modified version of the inverse transform is employed for the g component. This modified transform is constructed so that the original samples are placed at positions $(2i+1, 2j+1)$. In the modified (inverse)

transform, the first three lifting steps are defined as

$$\begin{aligned} y[2n] &\leftarrow y[2n] - y[2n+1] \\ y[2n+1] &\leftarrow y[2n+1] + y[2n] \\ y[2n] &\leftarrow y[2n] - y[2n+1]. \end{aligned} \tag{7}$$

It is straightforward to verify that these three steps swap the samples at odd and even positions (with a sign change). Specifically, after these three steps, $y[2n+1]$ is equal to the original value of $y[2n]$, while $y[2n]$ is equal to the original value of $-y[2n+1]$. The next two steps are the same as those employed by the standard 5/3 inverse transform, but update first the samples at even positions and then odd positions (rather than first odd and then even, as in the standard transform):

$$\begin{aligned} y[2n+1] &\leftarrow y[2n+1] - \left\lfloor \frac{y[2n] + y[2n+2]}{4} + \frac{1}{2} \right\rfloor \\ y[2n] &\leftarrow y[2n] - \left\lfloor -\frac{y[2n-1] + y[2n+1]}{2} + \frac{1}{2} \right\rfloor \\ &\approx y[2n] + \frac{y[2n-1] + y[2n+1]}{2}. \end{aligned} \tag{8}$$

Again, neglecting rounding, the resulting interpolator is depicted in Fig. 12. As before, the samples at odd positions are zero by construction and $x[i]$ denotes the i-th original sample before interpolation. As desired, the original samples appear at odd positions in the output, while the even positions contain the interpolated values.

It is worth noting that the JPEG 2000 standard requires the first lifting step of any *forward* transform –i.e., the last step of the corresponding *inverse* transform– to update odd positions. Therefore, a null (do nothing) step is included at the end of the lifting network for the inverse transform in Fig. 12. The following parameters can be employed with Kakadu `kdu_compress` to apply the modified DWT to the g component:

```
Ckernels=ATK
Catk:C2=3
Kreversible:I3=yes
Kextension:I3=SYM
Ksteps:I3={0,0,0,0},{2,-1,1,1},{2,0,2,2},
          {1,0,0,0},{1,0,0,0},{1,0,0,0}
Kcoeffs:I3=-0.5,-0.5,0.25,0.25,1,-1,1
```

REFERENCES

[1] B. Bayer, "Color imaging array," US Patent 3,971,065, Jul., 1976.

[2] K. Hirakawa and T. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 360–369, Mar. 2005.

[3] L. Fang, O. C. Au, Y. Chen, A. K. Katsaggelos, H. Wang, and X. Wen, "Joint Demosaicing and Subpixel-Based Down-Sampling for Bayer Images: A Fast Frequency-Domain Analysis Approach," *IEEE Trans. Multimedia*, vol. 14, no. 4, pp. 1359–1369, Aug. 2012.

[4] G. Jeon and E. Dubois, "Demosaicking of noisy Bayer-sampled color images with least-squares luma-chroma demultiplexing and noise level estimation." *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 146–56, Jan. 2013.

[5] J. Chang, H. Kang, and M. G. Kang, "Correction of axial and lateral chromatic aberration with false color filtering." *IEEE Trans. Image Process.*, vol. 22, no. 3, pp. 1186–98, Mar. 2013.

[6] Y.-C. Fan, Y.-F. Chiang, and Y.-T. Hsieh, "Constant-Hue-Based Color Filter Array Demosaicking Sensor for Digital Still Camera Implementation," *IEEE Sensors J.*, vol. 13, no. 7, pp. 2586–2594, Jul. 2013.

[7] X. Chen, G. Jeon, and J. Jeong, "Voting-Based Directional Interpolation Method and Its Application to Still Color Image Demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[8] J. Duran and A. Buades, "Self-similarity and Spectral Correlation Adaptive Algorithm for Color Demosaicking." *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 4031–4040, Jul. 2014.

[9] J. T. Korneliussen and K. Hirakawa, "Camera processing with chromatic aberration." *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4539–52, Oct. 2014.

[10] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, "Joint Demosaicing and Denoising via Learned Nonparametric Random Fields." *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 4968–4981, Dec. 2014.

[11] Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, "Adaptive residual interpolation for color image demosaicking," in *IEEE International Conference on Image Processing (ICIP)*, Sep. 2015, pp. 3861–3865.

[12] X. Chen, L. He, G. Jeon, and J. Jeong, "Multidirectional Weighted Interpolation and Refinement Method for Bayer Pattern CFA Demosaicking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 8, pp. 1271–1282, Aug. 2015.

[13] H. Siddiqui, K. Atanassov, and S. Goma, "Hardware-friendly universal demosaick using non-iterative map reconstruction," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 1794–1798.

[14] C. Zhang, Y. Li, J. Wang, and P. Hao, "Universal Demosaicking of Color Filter Arrays," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5173–5186, Nov 2016.

[15] *Digital Negative (DNG) Specification*, Adobe Std., Rev. 1.4.0.0, June 2012.

[16] D. Coffin. DCRAW - Dave Coffin's raw photo decoder. [Online]. Available: http://www.cybercom.net/~dcoffin/dcraw/

[17] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. Springer, 1993.

[18] *JPEG-LS*, ISO/IEC Std. IS 14 495-1, 14 495-2, 1998.

[19] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards and practice*. Kluwer Academic Publishers, Boston, 2002.

[20] T. Toi, M. Ohta, S. Systems, and I. Technology, "A subband coding technique for image compression in single CCD cameras," *IEEE Trans. Consum. Electron.*, vol. 45, no. 1, pp. 176–180, 1999.

[21] N. Zhang and X. Wu, "Lossless Compression of Color Mosaic Images," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1379–1388, 2006.

[22] S.-Y. Lee and A. Ortega, "A novel approach of image compression in digital cameras with a Bayer color filter array," in *Proceedings of the IEEE International Conference on Image Processing*, 2001, pp. 482–485.

[23] C. C. Koh, J. Mukherjee, and S. K. Mitra, "New Efficient Methods of Image Compression in Digital Cameras with Color Filter Array," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1448–1456, 2003.

[24] A. Bazhyna, K. Egiazarian, S. K. Mitra, and C. C. Koh, "A Lossy Compression Algorithm for Bayer Pattern Color Filter Array Data," in *International Symposium on Signals, Circuits and Systems*, vol. 2, Jul. 2007, pp. 1–4.

[25] H. Cuce, A. Cetin, and M. Davey, "Compression of Images in CFA Format," in *2006 International Conference on Image Processing*, 2006, pp. 1141–1144.

[26] X. Xie, G. Li, X. Li, Z. Wang, C. Zhang, D. Li, and L. Zhang, "A New Approach for Near-lossless and Lossless Image Compression with Bayer Color Filter Arrays," in *Proceedings of the Third International Conference on Image and Graphics (ICIG)*, 2004, pp. 1–4.

[27] A. Bazhyna and K. Egiazarian, "Lossless and near lossless compression of real color filter array data," *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1492–1500, 2008.

[28] S. Kim and N. I. Cho, "Lossless Compression of Color Filter Array Images by Hierarchical Prediction and Context Modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 1040–1046, Jun. 2014.

[29] E. Atsumi, "Digital camera system built on JPEG2000 compression and decompression," *SPIE Proceedings*, vol. 5017, pp. 254–262, 2003.

[30] S. Battiato, A. R. Bruna, A. Buemi, and A. Castorina, "Analysis and Characterization of JPEG 2000 Standard for Imaging Devices," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 773–779, 2003.

[31] B. Parrein, M. Tarin, and P. Horain, "Demosaicking and JPEG2000 compression of microscopy images," in *Proceedings of the International Conference on Image Processing*, 2004, pp. 521–524.

[32] R. Lukac and K. Plataniotis, "Single-Sensor Camera Image Compression," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 299–307, May 2006.

[33] S. Battiato, A. Buemi, L. D. Torre, and A. Vitali, "A fast vector quantization engine for CFA data compression," in *Proceedings of IEEE-EURASIP Workshop on Non Linear Signal and Image Processing (NSIP)*, 2003.

[34] A. Bruna, F. Vella, A. Buemi, and S. Curti, "Predictive differential modulation for CFA compression," in *Proceedings of the 6th Nordic Signal Processing Symposium (NORSIG)*, 2004, pp. 101–104.

[35] N.-X. Lian, L. Chang, V. Zagorodnov, and Y.-P. Tan, "Reversing demosaicking and compression in color filter array image processing: performance analysis and modeling." *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3261–78, Nov. 2006.

[36] K.-h. Chung and Y.-h. Chan, "A Lossless Compression Scheme for Bayer Color Filter Array Images," *IEEE Trans. Image Process.*, vol. 17, no. 2, pp. 134–144, 2008.

[37] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: a systematic survey," *SPIE Proceedings*, vol. 6822, pp. 68 221J–1–68 221J–15, 2008.

[38] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta. A Standard Default Color Space for the Internet: sRGB. November 1996. [Online]. Available: http://www.color.org/srgb.xalter

[39] *JPEG 2000 image coding system: Extensions*, ISO/IEC Std. 15 444-2, 2004.

[40] *Image technology colour management – Architecture, profile format and data structure*, International Color Consortium (ICC) Std. ICC.1:2010-12, 2010. [Online]. Available: http://www.color.org/icc_specs2.xalter

[41] Kakadu JPEG2000 software. [Online]. Available: http://www.kakadusoftware.com

[42] *JBIG2*, ISO/IEC Std. IS 11 544, 1993.

[43] ISO/IEC and ITU-T, "High Efficiency Video Coding (HEVC) HM reference software." [Online]. Available: http://hevc.hhi.fraunhofer.de

[44] *JPEG XT*, ISO/IEC Std. 18 477, 2014.

[45] Y. T. Tsai, "Color Image Compression for Single-Chip Cameras," *IEEE Transactions on Electron Devices*, vol. 38, no. 5, pp. 1226–1232, 1991.

[46] *Information technology - JPEG 2000 image coding system - Part 9: Interactivity tools, APIs and protocols*, ISO/IEC Std. 15 444-9, Dec. 2005.

[47] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.