Bayer CFA Pattern Compression With JPEG XS

Thomas Richter, *Member, IEEE*, Siegfried Fößel, *Senior Member, IEEE*, Antonin Descampe, and Gaël Rouvroy

***Abstract*— While traditional image compression algorithms take a full three-component color representation of an image as input, capturing of such images is done in many applications with Bayer CFA pattern sensors that provide only a single color information per sensor element and position. In order to avoid additional complexity at the encoder side, such CFA pattern images can be compressed directly without prior conversion to a full color image. In this paper, we describe a recent activity of the JPEG committee (ISO SC 29 WG 1) to develop such a compression algorithm in the framework of JPEG XS. It turns out that it is important to understand the "development process" from CFA patterns to full color images in order to optimize the image quality of such a compression algorithm, which we will also describe shortly. We introduce (1) a novel decorrelation step upfront processing (the so-called Star-Tetrix transform), along with (2) a pre-emphasis function to improve the compression efficiency of the subsequent compression algorithm (here, JPEG XS). Our experiments clearly indicate a gain over a RGB compression workflow in terms of complexity and quality (between 1.5dB and more than 4dB depending on the target bitrate). A comparison is also made with other state-of-the-art CFA compression techniques.**

***Index Terms*— Image coding, JPEG XS, CFA pattern compression.**

I. Introduction

DIGITAL images are today typically acquired by color filter array (CFA) pattern sensors, that is, by a rectangular sensor array where each sensor element is covered by one out of multiple color filters. A typical arrangement is that of a Bayer pattern [1] of a repeating 2×2 grid matrix containing one red, one blue and two green sensitive sensor elements, repeating over the entire grid. To reconstruct a full-scale full color image from such a source, often called demosaicking, it is therefore necessary to interpolate missing sample values at all sample positions, a problem for which multiple algorithms have been developed over time [10]. However, as we will see, interpolation and upsampling are only one out of many steps that are necessary to reconstruct an observable image, and we will describe these steps briefly in section III as it is important to consider them in an image compression algorithm targeting CFA pattern sourced images.

Manuscript received October 28, 2020; revised April 9, 2021 and May 28, 2021; accepted June 29, 2021. Date of publication July 16, 2021; date of current version July 22, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charith Abhayaratne. *(Corresponding author: Thomas Richter.)*

Thomas Richter and Siegfried Fößel are with the Fraunhofer IIS, 91058 Erlangen, Germany (e-mail: thomas.richter@iis.fraunhofer.de).

Antonin Descampe is with UCLouvain, 1348 Louvain-la-Neuve, Belgium.

Gaël Rouvroy is with intoPIX, 1435 Mont-Saint-Guibert, Belgium.

Digital Object Identifier 10.1109/TIP.2021.3095421

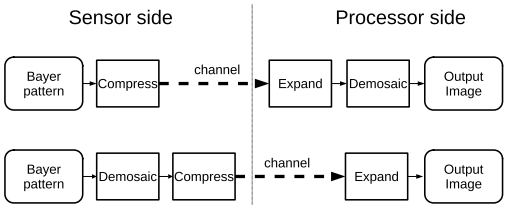

Fig. 1. Complexity reduction by compressing CFA patterns. Top: the approach discussed in this paper. Bottom: The traditional approach.

Transport and compression of images are part of many industrial applications. Machine vision systems for example are equipped with multiple CFA pattern sensor cameras that communicate over various industrial bus systems, like GigE Vision, CameraLink, USB3.0 Vision [2] or CoaXPress [3], with a powerful image processing computer system to test manufactured units for quality. Modern cars are equipped with multiple CFA pattern sensors that communicate over automotive ethernet with an Electronic Control Unit (ECU) to implement various driver assistant functions or to visualize surround camera views on the driver's screen. As the bandwidth of industrial bus systems is limited, but the image resolution of sensors keeps increasing, it becomes attractive to compress the source signals near the sensor prior transporting them over a bus. In addition, as the upscaling and interpolation process to a full scale color image is resource-hungry, we instead want to consider applications where the CFA source is compressed directly close to the sensor, and the "development" of this source to a full scale color image happens at the display unit where more processing power or a graphical processing unit (GPU) is available.

Fig. 1 compares two setups when an image accquired by a CFA pattern sensor is to be transported over a limited bandwidth channel. The first setup at the top of the figure compresses the CFA pattern directly as discussed in this work, the second setup follows the traditional approach which first converts the CFA pattern to an RGB image and then compresses this image afterwards. It seems evident that the sensor-side complexity of the top setup is smaller as it moves the demosaicking step to the decoder, but we will also provide experimental evidence on this in section VII.

To optimize for this particular use case, it is important to understand how we define and measure *image quality* in this framework; it turns out that it is insufficient to measure PSNR in the CFA pattern domain and a more careful process must be utilized instead. We will describe this process in more detail in section IV. Note that the proposed JPEG XS extension

operates on CFA pattern images entirely, i.e. the input of the compressor and the output of the decompressor are CFA pattern images.

Once image quality is understood, we describe in section V the processing steps of a JPEG XS [15]–[17] extension currently under standardization that applies to such source images, give a brief review of its algorithm and its history, and then describe those processing steps in more detail that are relevant for CFA pattern compression. We will provide evidence that it is important to take all steps of the development algorithm from a CFA pattern to a full scale color image into account to achieve optimal compression results; in particular, only considering the sensor element arrangement as done in other works and the upscaling algorithm is not sufficient.

In section VII we describe experiments and provide experimental results that show how far each of the steps of the compression pipeline contributes to the overall quality. As to be seen, an overall improvement of about 2dB can be gained over a naive approach.

II. Related Work

It was already observed by Zhang and Wu in [4] that compressing the CFA pattern without first converting it to a three-component RGB image could improve compression performance and reduce the complexity. In their work, they describe two compression methods, one that first performs bi-linear interpolation on the two green channels, and then code the red/green and blue/green differences, and a second where the Bayer pattern is coded directly with a wavelet, noting that the first horizontal and vertical highpasses then reproduce a color-difference filter. A simple lossless Golomb-Rice coder is also introduced. Note that bi-linear interpolation of green channels and computing the color difference is *identical* to the first lifting step of the Star-Tetrix transformation, see eqn. (5) and Fig.5.

In the framework of JPEG XR, Malvar and Sullivan [11], [12] proposes to consider each 2×2 super-pixel element of a Bayer type CFA pattern as one sample of a four-color image, and proposes a color decorrelation transformation that is derived from the YCgCo transformation to be applied to such superpixels. That is, each 2×2 matrix element is considered one sample of a four-component image having half the width and half the height of the sensor, and each of the four elements of this matrix make up one component of it, see Fig. 2.

S. Mohammed and K. Wahid describe in [5], [6] a one-dimensional and two-dimensional lossless colour decorrelation process for CFA pattern images operating on 2×2 superpixels as in [11]. The one-dimensional transformation is identical to a horizontal Haar filter on the green/blue or green/red lines, an optional vertical filtering step uses a second Haar transform on the average output of the horizontal filter. The combined horizontal and vertical filter can be further simplified in a two-dimensional transformation.

In [7], [8], Suzuki improved the results of [12] by extending the color decorrelation transformation beyond superpixel boundaries such that it includes also a spatial decorrelation.

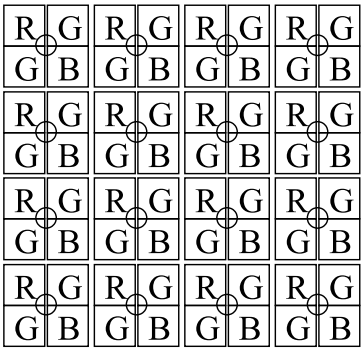

Fig. 2. A CFA pattern image, each circle indicates a sample grid point of a four component image, the four components indicated as squares. Typically, as seen in the figure, there are one red, one blue and two green channels.

Instead of confining the transformation to 2×2 superpixels, the proposed transformation there accesses neighbouring samples regardless to which superpixel they belong to. Similar to the works of Malvar, the transformation proposed here is also based on the YCgCo transformation, or is rather an extension of it.

M. Hernández-Cabronero et al describe in [9] a lossy and lossless CFA pattern compression based on JPEG 2000 which also allows simple reconstruction to an RGB image by merging parts of the demosaicking steps into the JPEG 2000 wavelet stage, creating a filter that is approximately identical to bi-linear interpolation. Color space conversion and conversion to a non-linear (gamma-corrected) colorspace are expressed by means of an ICC profile which is represented in the JPEG 2000 file format, though compression is performed in the linear domain. The color decorrelation transformation from eqn. (2) of section III.B of their work is up to rounding identical to the transformation (7) and is also included in our experimental validation.

This work is an extension of [31], [32] of the same authors. First of all, we consider a variant of the reversible color transformation (RCT) introduced in the framework of JPEG 2000 [23], [24], [26]. This variant of the RCT transformation was already discussed in [31], but we will briefly introduce it again in section VII. The work [32] replaced it by the better performing Star-Tetrix transformation, to be discussed in section VI, with experiments comparing it with YCgCo in section VII.

In this work, we note that processing of CFA pattern sources to full resolution color images includes additional steps whose knowledge can improve the performance of an image compression algorithm significantly. These steps include gain corrections, color space conversion steps, as well as the transformation to a gamma-corrected (non-linear) space more suitable for observation, display and processing. It is important to observe that an additional *nonlinear* transformation, i.e. a pre-emphasis and post-emphasis function can improve the performance significantly, see section VII, and that previous works did not include such a step as only lossless algorithms were considered there. While it was already noted in [31] that the knowledge of the camera white balance can improve the compression performance, only the rate allocation process

was modified there. In this work, the linear decorrelation transformation itself includes white balance correction factors.

In particular for lossless compression of CFA pattern data, data-dependent transformations based on context modeling have also been proposed [13], [14]. There, cross-component correlations are removed by data-dependent linear predictors that are selected according to the local neighbourhood of a pixel in the CFA pattern. Lossless compression performance reported there is approximately 2 : 1 on 8 bit CFA pattern images, while our test image set (see section VII) consists, however, of 14 bit CFA pattern input. When disabling the non-linear pre-emphasis functions, the remaining CFA coding tools of JPEG XS can also operate in lossless mode and we observe an average compression ratio of 1.6 : 1 on it. The compression ratio increases to 2.4 : 1 when disregarding least significant bits by downscaling the input data first to 8 bits, i.e. in a situation that is roughly comparable to [14]. While the method proposed in this paper can also operate in lossless mode, our focus is on lossy compression, or scalable lossy-to-lossless compression. Obviously, lossy compression can reach much higher compression factors, and our lowest test point is 1bpp, corresponding to a 14:1 compression. Note that data-dependent methods such as those discussed in the above works are unsuitable for lossy compression as the prediction context may change after quantization and thus a decoder cannot invert the encoder-side context decisions anymore.

In the next section, we will discuss the whole processing pipeline of CFA pattern images to human observable RGB images in more detail.

III. CFA Pattern Processing

The pipeline of converting a CFA pattern image to a full resolution color image consists of multiple processing steps beyond interpolation of sample values. An initial step often realized is denoising in the CFA domain, for example by a wavelet filter and deadzone quantization that removes low-amplitude signals [18], [19].

This step is followed by *black level removal*, which consists of the subtraction of an offset value from the sample value and clamping all sample values below 0 to 0. It suppresses sensor noise in black areas, and also suppresses sensor values due to flare light reaching the sensor by reflections on the interior of the camera case. The black level, also denoted as *optical black* or *OB value* is sensor and camera dependent, but does typically not depend on the color channel. In some cases, the optical black is already removed by the camera electronics.

Black level removal is followed by *gain correction* which adjusts the sampled signal across channels for the unequal sensitivities of the sensor elements. In particular, the actual sensor is not equally sensitive to all incoming wavelengths, and as the color filter on top of the element removes fractions of the spectrum, the sensitivity of the overall sensor-filter element is the point-wise product of the sensor sensitivity times the wavelength dependent filter characteristics. This unequal sensitivity is compensated by multiplying the black-level corrected signal by a color-channel dependent gain factor. If the input vector of this correction is written as four-dimensional column vector, this step can be understood as the multiplication with a diagonal matrix. Similar to the first step, the gain factors are camera dependent and typically embedded in the "raw" format written by the camera, e.g. TIFF-EP [20], DNG or a vendor-specific format.

A conversion to an intermediate colorspace comes next. This conversion consists again of a multiplication of the sample-column vector with a diagonal matrix, though the matrix elements are now scene, and in particular illumination-dependent. They transform the signal to an illumination independent intermediate color space and thus implement a simple form of white-balance correction. By which means the camera selects appropriate gain factors, and hence the white balance correction is vendor specific and not documented, though the matrix elements are part of the vendor specific file format in which raw signals are recorded. For the sake of simplicity, and in an abuse of language, we will denote the product of the gain matrix and the white-balance matrix simply as *white balance correction*, even though it consists of two contribution.

At this stage, the image still consists of one sample per sensor element; however, a full resolution color image requires 3 color components per sample position, making an upsampling step necessary. The simplest possible algorithm is bi-linear interpolation of the existing signals to all sample positions; it is, however, of little practical value as it creates interpolation artefacts especially around horizontal and vertical image edges due to the alternating color filters along them, also known under the term *zipper defects*. More sophisticated algorithms such as the AHD [10] algorithm attempt to identify the orientation of edges and then perform filtering along the direction of the edge.

The output of the previous stage is a full-resolution color image, but it is still in an intermediate color space, and most notably, sample values are proportional to the light intensity at the sensor. Many applications of digital images require, however, gamma corrected signals, or in general, processing by a non-linear pre-emphasis function that corresponds to the non-linear sensitivity curve of the human visual system. Such representations allocate less code space volume to bright pixels than to dark pixels, corresponding to the (approximately) logarithmic eye sensitivity. Typical representations are generated using transfer characteristic curves from sRGB [21] or BT.2020 [22]. Accordingly, the last step consists first of a conversion from the linear intermediate camera color space to a coordinate system relative to the primaries of the target color space, followed by a color-space specific non-linearity. In this work, we will focus on the sRGB non-linearity only.

It turns out that it is important to consider not only the interpolation, but also the non-linearity and the white-balance correction for high-quality CFA pattern compression algorithms. What exactly we mean by *quality* will be discussed in the next section.

IV. Quality of CFA Pattern Images

For optimizing the efficiency of a CFA pattern compression algorithm, it is important to define the term *image quality* in correspondance with the application target of the algorithm;

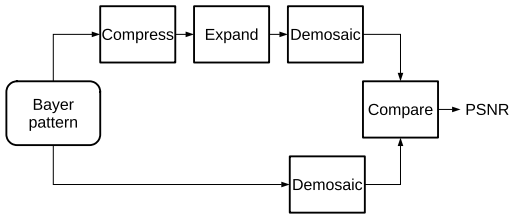

Fig. 3. The measurement workflow: The full-scale RGB image after demosaicking is compared with demosaicking with compression in the loop.

while this seems obvious, a particular choice may turn out ideal for one application, but suboptimal for another.

For the purpose of this work, we define image quality as the output of an image quality index *after* conversion ("development") of a CFA pattern into a full-scale sRGB color image, i.e. we define image quality by comparing the output of the demosaicking step including and excluding lossy compression in the CFA pattern domain; in particular, the input to the proposed compression algorithm is an unaltered camera-raw image consisting of one sample per sensor element. We therefore follow the workflow of Fig. 3:

A source CFA pattern is converted to a full scale color image without any modification, providing a reference image. The same CFA source is then compressed to a particular bitrate, then expanded to an (altered) CFA pattern image, and this output CFA image is also converted to a full scale color image. This second image will contain distortions due to the compression algorithm, and these two full scale color images are then compared by a quality index. For the sake of simplicity, we restrict ourselves here to PSNR, though appreciate that there are of course choices that correlate better to human perception.

Note that this definition of quality may order the compression efficiencies of two candidate compression algorithms differently than comparing the PSNR value of the CFA pattern itself. In fact, an algorithm that minimizes the PSNR in the CFA domain may not work well in the full scale color image domain like RGB, and vice versa. This definition also implies that the quality evaluation necessarily includes non-linear elements, namely the upsampling conversion and the conversion to a non-linear, gamma corrected output color space.

This quality definition also allows us to evaluate in how far compression in the CFA domain is competitive to other approaches, such as first generating a full scale color image from the CFA source and compress this image instead. That is, it allows us to compare compression in the CFA domain with compression in the color image domain. Section VII will provide results on this question.

For our experiments, we implement the above algorithm by means of the open source program `dcraw` which supports manifold CFA formats: To generate a full scale color image from a CFA output, we run `dcraw` directly on the source. A small modification allows us to alter the bit precision of the target color space between 8 and 16 bits; in our test, we use 10 bit output consistently, but any other precision would be possible as well.

To generate an input suitable for a compression algorithm, we run `dcraw` in "document mode" which gives a 16-bit, single component grey scale image in which each sensor element is represented by a single sample. While the sample values are represented in a 16-bit container, the actual camera precision is typically below 16 bit, in our experiments only 14 bit. Thus, we need to modify the signalled bit precision of the container format accordingly. This step does not alter the sample values, it just corrects the meta-information of the container format to match the precision of the camera.

The resulting grey-scale image is then compressed with a candidate algorithm to a target bitrate, which we adjust between 1 and 6 bits per sample. Note that a "sample" corresponds here to one sample of the CFA pattern, not one sample consisting of three channels of a full color image. The compressed bitstream is then expanded again. A modification of `dcraw` allows us now to replace the raw data coming from the camera file with the compression result while preserving all the vendor specific metadata from the original, and thus allows us to convert the compression result to a full scale color image using the same metadata as the original file. Thus, `dcraw` is here used in a "black box" approach hiding all camera and vendor specific operations.

The two images — the full scale color image derived from the original input, and the full scale color image generated by substituting the sample values with those of the compression result — are then compared by means of PSNR. These numbers we will report in section VII.

V. JPEG XS

In this section, we will briefly report on the coding engine of the JPEG XS standard in general before going into the adaptions we made for CFA pattern compression. For a more comprehensive introduction, the reader is referred to [15]–[17].

JPEG XS is a wavelet based image compression codec, though unlike JPEG 2000 [24], has been designed with high-speed, low-latency low-complexity considerations in mind. The compressor first applies a component decorrelation transformation that is identical to the RCT of JPEG 2000 for regular RGB images, though this step is for our purpose replaced by a non-linear point transform and the so-called *Star-Tetrix* component decorrelation transformation. We will describe this transformation in more details in section VI.

Following this step, the color-decorrelated samples are wavelet transformed by the LeGall 5/3 wavelet, using 5 horizontal and 0 to 2 vertical levels. This somewhat asymmetrical setup ensures that the end-to-end latency of the codec remains low, 32 lines for regular RGB images and 40 lines for CFA images.

Wavelet coefficients are then pre-quantized to a 16-bit sign-magnitude representation which is the input to the rate-allocation step, which decides on the (final) quantizer bucket size and the entropy coding mode used to encode the data. Quantization is either a deadzone quantizer that cuts off the T least significant bits of the pre-quantized input, or a data-dependent uniform quantizer derived from the deadzone quantizer using the same number of buckets. The value of T

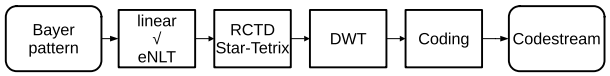

Fig. 4. Encoding steps of the algorithm discussed in this paper. All steps are reverted at the decoder whose output is again a CFA pattern which is a lossy version of the original input.

is derived from a locally selected quantization strength and the wavelet filter gain of the band being quantized. If sufficient data volume remains available, selected bands may receive one additional bitplane in an order that is dependent on the fractional part of the wavelet gain not expressible by the integer T value. This dynamic assignment of refinement bitplanes compensates for the otherwise coarse quantization steps implied by bitplane truncation.

For entropy coding, quantized wavelet coefficients are collected into groups of four coefficients each, so called *coding groups*. The number of populated bitplanes in each coding group is called the *bitplane count*. Entropy coding proceeds in four coding passes: First, significance coding encodes in a single bit whether groups of eight consecutive coding groups have a bitplane count of zero, and thus do not require any further coding. In the second coding pass, the remaining bitplane counts are encoded, either by a differential unary code relative to the line above, a unary code without prediction, or in a fall-back mode that encodes the count in a fixed length code of four bits each. The latter mode guarantees that the entropy coded size of the bitplane count does not exceed an easy to compute upper bound. The purpose of this raw coding mode is to ensure that the size of decoder-side buffers is bounded from above by an easily computable bound.

Following the bitplane count encoding, the magnitude of the wavelet coefficients is written directly to the output, four bits per coding group per bitplane, without further encoding. In the last step, the signs of non-zero coefficients are written out.

Albeit this coding mechanism is simple — only the bitplane counts are entropy coded — our experiments [15] have shown that it is competitive in the bitrate regime JPEG XS has been designed for, namely high-quality visually lossless coding. More details on the experiments validating JPEG XS can be found in [16].

VI. Modifications for CFA Compression

While wavelet transformation, quantization and entropy coding remain unaltered, the steps upfront the spatial decorrelation transformation implemented by the wavelet filter require significant changes to address the peculiarities of CFA pattern data. It is important to note that the processing steps introduced in this section do not convert CFA data with one sample value per sensor element to the RGB domain with 3 sample values per sample grid point, but rather implement tools that improve compression performance operating on the CFA data as-is. The overall steps taken at the encoder are depicted in Fig. 4, with several options for various steps evaluated in section VII. Note that in our approach, the decoder implements the inverse processing steps and thus reconstructs a codestream to a CFA pattern.

First of all, it should be noted that CFA sensor values are proportional to the incoming light intensity, whereas full resolution color images developed from such data use gamma-corrected or other forms of non-linear color spaces. That is, the final output of a CFA decompression and demosaicking algorithm undergoes a non-linear transformation before becoming an observable image, see section III. Note again that this conversion to an observable image happens outside of the compression/decompression cycle. The non-linearity in this conversion process implies that the quantizer within such a CFA compression algorithm can be understood to have effectively non-equally sized buckets, which is known to be not ideal in the high-bitrate approximation [27]. To compensate for this effect, our CFA compression pipeline contains an approximation of a typical non-linearty used for development of CFA data, i.e. its conversion to a full-scale RGB image.

This non-linearity consists of a linear ramp of a slope that is a power of 2 between the optical black value (see section III) and the toe-threshold of the sRGB color space, followed by a square root (encoder side) or square function (decoder side) above the toe threshold, and an inverted square root or square below the optical black. In particular, one has as encoding non-linearity

$$f(x) := \begin{cases} b_1 - \sqrt{a_1 - x} & \text{for } x < \Theta_1 \\ (x - b_2) \cdot 2^{-e} & \text{for } \Theta_1 \leq x < \Theta_2 \\ b_3 + \sqrt{x - a_3} & \text{for } x \geq \Theta_2 \end{cases} \tag{1}$$

where Θ_1 is the optical black level and Θ_2 the toe-threshold of the gamma correction. The constants a_1, b_1, b_2, a_3, b_3 can be derived from Θ_1 and Θ_2 and the constraints that $f(0) = 0$ and f being continuously differentiable. The codestream neither contains any of the a_i or b_i constants nor the thresholds Θ_1 or Θ_2, but the decoder-side decision points $T_1 = f(\Theta_1)$ and $T_2 = f(\theta_2)$, as well as the exponent value e. Note that no output scaling is performed within f to keep the number of multiplications low. This typically results in the effect of reducing the dynamic range of the output slightly, but has the benefit of creating additional headroom for overshoots in the wavelet transformation.

Note that we do not use the 2.2 exponent of the sRGB (or related) color spaces due to the implementation complexity of this operation as it would either require transcendental math functions or a large lookup table, though JPEG XS is targeted at high-speed CPU and low-cost FPGA implementations. Instead, it is approximated by a power of 2 which is trivial to implement at the decoder. The inverted square root below optical black ensures that the increment in dynamic range compared to an all-linear pre-emphasis remains small. Note that we attempt to also preserve values below the optical black even though the development of the CFA pattern to a full scale color image may remove them. Some algorithms operating in the CFA domain may require such sample values, e.g. advanced denoising operations such as [18]. While it sounds curious that it is necessary to preserve values below the optical black, section VII provides evidence on a workflow containing such a denoising algorithm.

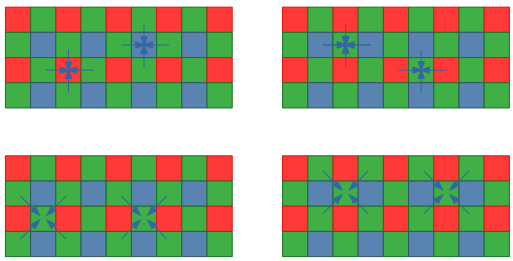

Fig. 5. The four lifting steps of the Star-Tetrix transformation, from left to right, top to bottom: Compute C_b,C_r, compute luminances, compute the Δ luminance difference, compute the average luminance.

Particular care needs to be taken to avoid any overflow during the computation of the post-emphasis f^{-1} at the decoder side, as it contains two squaring operations whose input may exceed the bitrange of a typical 18×18 bit hardware multiplier due to the Gibbs phenomenon:

$$f^{-1}(x) = \begin{cases} a_1 - (b_1 - x)^2 & \text{for } x < T_1 \\ 2^e x - b_2 & \text{for } T_1 \le x < T_2 \\ a_3 + (x - b_3)^2 & \text{for } x \ge T_2 \end{cases} \tag{2}$$

To avoid such an overflow, the standard mandates to clamp the input of the multiplier to 18-bit precision, i.e. clamp $x-b_3$. A short computation shows that b_3 is always non-negative and thus clamping at this stage is less intrusive than clamping x directly. For that, one first verifies that

$$b_3 = 2^e \left[\Theta_2 - \left(\sqrt{\epsilon^2 + \Theta_1} - \epsilon \right)^2 \right] \text{ with } \epsilon := 2^{-e-1} \tag{3}$$

to ensure that f is continuously differentiable. It remains to be seen that the square bracket is non-negative, or equivalently, that

$$\Theta_2 \ge \left(\sqrt{\Theta_1 + \epsilon^2} - \epsilon \right)^2 \tag{4}$$

Now consider the following term:

$$\Theta_1 - \left(\sqrt{\Theta_1 + \epsilon^2} - \epsilon \right)^2 = \epsilon^2 \left(\sqrt{\frac{\Theta_1}{\epsilon^2} + 1} - 1 \right) \ge 0$$

Thus Θ_1 is larger or equal to the term in the brackets. Since $\Theta_2 \ge \Theta_1$, eqn. (4) holds, and thus $b_3 \ge 0$.

The multi-component decorrelation transformation following the non-linearity combines both a spatial and a color component similar to the transformations proposed by Suzuki [7], [8], though are derived from the JPEG 2000 RTC transformation. This multi-decomponent decorrelation transformation should not be confused with a demosaicking algorithm; it is only a tool to improve compression efficiency, but does not — unlike a demosaicking algorithm — create 3 components per image sampling grid point. Section VII will compare the Suzuki transformation with the one proposed here in more detail.

The *star tetrix* transformation is fully invertible, and consists of four lifting steps, cf. Fig. 5: In the first lifting step, the average of the four surrounding green samples is subtracted from each red and each blue sample value, resulting in approximate C_r and C_b coordinates. This step is a lookalike of the first two lifting steps of the JPEG 2000 RCT transformation, except that the green channel at red and blue sample positions is bilinearly interpolated. It was already noted in [4] that the difference between interpolated colors is more amenable for coding. Denote by the super-indices l,r,t,b the sample position to the left, right, top or bottom of the current sample, then this first lifting step reads:

$$\begin{aligned} C_b &:= B - \left\lfloor \frac{G^l + G^r + G^t + G^b}{4} \right\rfloor \\ C_r &:= R - \left\lfloor \frac{G^l + G^r + G^t + G^b}{4} \right\rfloor \end{aligned} \tag{5}$$

Sample positions that lie outside of the sampling grid are reflected back inwards to give this expression a meaning at the boundary of the grid.

The second step adds a weighted average of the C_r and C_b samples surrounding each green sample. The weights are always powers of two, but their selection depends on the white-balance parameters of the camera. That is, it is the purpose of these weights to compensate for the conversion to an intermediate color space in the development of CFA pictures. The output of this step are two approximate luma channels. Using the same notation as above, this second lifting step reads:

$$\begin{aligned} Y_1 &:= G + \left\lfloor \frac{2^{w_r}(C_r^l + C_r^r) + 2^{w_b}(C_b^t + C_b^b)}{8} \right\rfloor \\ Y_2 &:= G + \left\lfloor \frac{2^{w_r}(C_r^t + C_r^b) + 2^{w_b}(C_b^l + C_b^r)}{8} \right\rfloor \end{aligned} \tag{6}$$

where w_b and w_r are non-negative integer white-balance constants. For $w_b = w_r = 0$, one finds by inserting eqn. (5) into eqn. (6) that $Y_{1,2} \approx (r_{av}+2\ g_{av} + b_{av})/4$, where r_{av}, g_{av} and b_{av} are average/filtered red, green and blue sample values. Thus, the two lifting steps above correspond to a conversion to YC_bC_r coordinates, with Y_1 and Y_2 being two luma values per super-pixel. The chroma exponents shift the white point closer to the weight point of the camera, improving the decorrelation efficiency, as seen in the experiments in section VII.

The next two steps now compute from the two luma values an average luma value $\bar{Y}$ and a luma difference value Δ. That is, a first spatial decorrelation transformation is already performed within the color decorrelation. First, compute luma differences by:

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

Then, update the remaining luma channel by Δ:

$$\bar{Y} := Y_2 + \left\lfloor \frac{\Delta^l + \Delta^r + \Delta^t + \Delta^b}{8} \right\rfloor$$

As above, one finds that $\bar{Y} \approx (Y_1 + Y_2)/2$.

The output of the *Star-Tetrix* transformation is a four-channel signal, consisting of luma, C_b, C_r and Δ. The first three channels are input to a wavelet transformation, the Δ channel — already consisting of a spatial difference signal —

is quantized and encoded as-is. It will be seen in section VII that skipping the spatial decorrelation of Δ is beneficial in terms of coding efficiency.

The transformation defined by these four steps is easily seen to be invertible, and to have a delay of four lines, i.e. one line per lifting step.

It is important to notice that these preprocessing steps go beyond those proposed by other authors, e.g. [8]: They mirror spatial filtering in the demosaicking step, color transformation to an intermediate space before filtering, as well as non-linear gamma transformation applied as final step. Thus, in a sense, our preprocessing steps are low-complexity approximations of similar steps performed in the development of CFA samples to full resolution color images. We will see in the next section that including such steps has a significant impact for the performance of the overall compression.

VII. Experiments

In this section, we evaluate the performance of the mentioned CFA compression workflow, along with some additional transformations found in the literature [7]. The evaluation workflow here follows the one proposed in section IV, i.e. we measure the PSNR in RGB space of a CFA pattern source relative to a constructed, recompressed CFA pattern. Additionally, in order to evaluate in how far CFA compression is beneficial, we also compare to a workflow with compression of the "developed" CFA pattern in RGB space, see Fig. 8 for the workflow.

A. Complexity

First, to demonstrate that compression in the CFA pattern domain is beneficial in terms of complexity, we measure the average running time of a JPEG XS compressor on CFA patterns on our dataset and compare it with the average running time of the same encoder on demosaicked images in the RGB domain and the running time of two demosaicking algorithms: Simple bi-linear interpolation, and AHD [10] interpolation. We also include the encoding time of the CFA pattern by JPEG 2000 to estimate the complexity of the solution proposed in [9], configured to a fast configuration: 5 decomposition levels, 1 layer, 5/3 lossy wavelet, bypass encoding, and an adaptive heuristic rate allocation that truncates bitplane coding early. The input to JPEG 2000 undergoes a pre-processing step that consists of a conversion of the CFA pattern to a four-component image, and a decorrelation transformation with the Reversible Color Transformation plus Delta (RCTD). This transformation computes an average green value and a green difference value and then computes the RCT of JPEG 2000 [24] on red, g_{av} and blue. The entire transformation thus reads

$$\begin{aligned} g_{av} &= \left\lfloor \frac{g_1 + g_2}{2} \right\rfloor \quad Y = \left\lfloor \frac{r + b + 2g_{av}}{4} \right\rfloor \\ C_b &= b - g_{av} \quad C_r = r - g_{av} \\ \Delta &= g_1 - g_2 \end{aligned} \tag{7}$$

It can be easily seen that this transformation is invertible without loss. This transformation is up to rounding conventions identical to that used in [9] section III.B, eqn. (2) and the RCT transformation implied by the following JPEG 2000 compression. In our measurements, the pre-processing step is not timed and does not add to the complexity of the JPEG 2000 encoder. The dataset consists of 15 CFA pattern images of 8288×5520 sensor elements with 14-bit precision and a black-level of 400, and was kindly provided by NIKON, see Fig. 6 for a subset of the images.

In our considered use-case of requiring to transmit an image captured by a CFA pattern type sensor over a limited bandwidth channel, see Fig. 1, a JPEG XS compressor near the sensor would replace a demoasicing algorithm followed by a traditional lossy image encoder there. The complexity at the processing unit implementing the decoder and demosaicking algorithm is less relevant; in our foreseen application scenario, the computer system at the decoder would also run image analysis tasks such as object detection or classification and would therefore be more powerful.

Figure 7 shows the result on an intel i7-4790 CPU at 3.60GHz, measuring execution time in seconds, over the target bitrate. The time spent in operating system calls such as to perform disk I/O is excluded, all data is brought into memory first and kept in memory. Time is acquired over ten cycles, with two "warm-up" cycles upfront to allow the operating system adjust the CPU clock speed and fill the CPU caches. Both the compressor and the demosaicking algorithm run here only on a single thread, though both are easy to parallelize to improve the performance. The demosaicking algorithm includes the full processing steps from a CFA pattern image to an image in the RGB colorspace, including black-level removal, white-balance adjustment, interpolation and conversion to the target colorspace. Note that in this experiment the black-level as well as the white balance parameters were derived from the metadata in the source image recorded by camera; only minimal steps, i.e. parsing the source format, are necessary to obtain them.

While all algorithms are also optimized from an algorithmic point of view, neither the JPEG XS implementation nor the JPEG 2000 implementation nor the demosaicking algorithm use SIMD instructions to employ data-parallelism of the intel processor. The JPEG XS implementation is that from Fraunhofer, the demosaicking algorithm comes from `dcraw`, a well-known open source implementation, the JPEG 2000 implementation is that from AccuSoft.

JPEG XS compression in the CFA domain is consistently faster than the bi-linear demosaicking alone, and more than 7 times as fast as the more advanced AHD algorithm. Also, compressing in the RGB domain with the same compressor is consistently slower than compressing the CFA pattern directly, and approximately of the same complexity as demosaicking by bi-linear interpolation. This is likely because the RGB compression engine has to operate on 3 times the source data compared to an encoder operating on a single-sample per sensor CFA pattern image. In applications, the complexity of the demosaicking algorithm and the follow-up image compression have to be added, cf. Fig. 1, and thus even with simple bi-linear

Fig. 6. Example images from the test set, the image have been generated by Nikon Corporation, and all intellectual property rights of the original images remain with Nikon Corporation. Test images were taken at the Botanical Gardens, Graduate School of Science, The University of Tokyo. Courtesy of Nikon. The images here have been already converted to the RGB domain, the experiments are performed on the corresponding CFA pattern data consisting of 15 images with 14-bit precision of 8288 × 5520 sensor elements each.

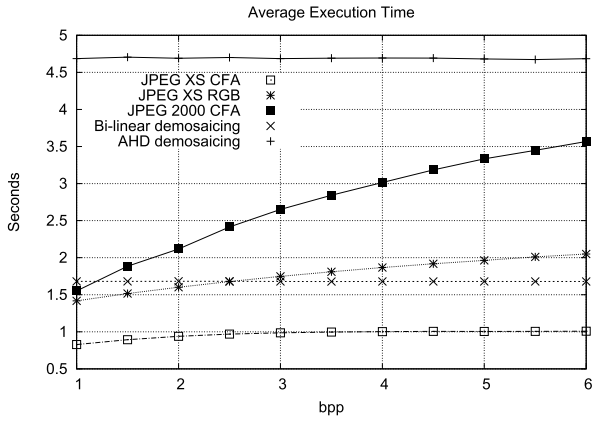

Fig. 7. Average running time in seconds of the JPEG XS encoder in the CFA domain and in the RGB domain compared to JPEG 2000 and that of two demosaicking algorithms implemented by `dcraw`, namely bi-linear interpolation and AHD interpolation. JPEG XS is run here in the best possible configuration, i.e. with Star-Tetrix transformation, white balance, extended non-linearity and 2 vertical decomposition levels. The graph "JPEG XS RGB" does *not* include the time for demosaicking the image before.

interpolation, complexity at the sensor side can be reduced by a factor of more than two by compressing the CFA data directly. Readers should also be aware that the faster, but simpler bilinear interpolation leads to sub-optimal quality and is prone to generate some defects — also known as "zipper artifacts" — that are rarely acceptable in professional applications. For details, we refer to [10]. The speed advantage is consistent over the entire bitrate range and is almost independent from the target bitrate. JPEG 2000 has for bitrates lower than 2bpp a moderate complexity that is roughly comparable to that of the bi-linear demosaicking algorithm, its throughput is however a lot more bitrate dependent than that of JPEG XS.

While in real-world applications, the actual codec or the demosaicking algorithm would be implemented in an FPGA or even an ASIC, the above measurements can be at least taken as an indication of the complexity of the various alternatives.

TABLE I

COMPLEXITY OF VARIOUS COLOR DECORRELATION TRANSFORMATIONS IN OPERATIONS PER 2 × 2 SUPER PIXEL. IN PARENTHESIS: OPTIONAL/ CONDITIONAL OPERATIONS

Transformation	Additions	Shifts	Square-Root
RCTD (Eqn. (4) and [24])	6	3	0
Zhang & Wu [4]	8	2	0
Mohammed & Wahid [5], [6]	6	3	0
YCgCoD [7], [8]	20	6	0
Star-Tetrix	24	6 + (2)	0
Linear	4	4	0
Square NLT	4	4	4
Extended NLT	12+(8)	4+(4)	(4)

Table I lists the number of elementary operations for the color decorrelation transformations discussed in section VII-D per 2 × 2 super-pixel. For the Star-Tetrix, 2 additional shifts are required in case white-balance gains are included in the transformation. The complexity of an additional non-linearity has to be added, it also includes the level-offset added to each sample value, and the bit-precision adjustment required to convert it to the JPEG XS internal precision. The extended non-linearity requires at most 8 (2 per sample) additional comparisons whose complexity is identical to that of an addition, and either a square root or a shift operation. When profiling our implementation in the best performing configuration (2 vertical levels, extended NLT, white balance enabled), the complexity of the preprocessing accounts for approximately 25% of the running time of the encoder at 3bpp.

B. Contribution of Coding Tools

Figure 9 shows the results of this experiment for various settings of the encoder, as PSNR differences relative to compression in RGB space. The source images are here taken from the same CFA source image dataset. The following procedure is used to obtain the chroma weights w_r and w_b of the Star-Tetrix transformation: By means of the open source `dcraw` program, the camera output is analyzed and the

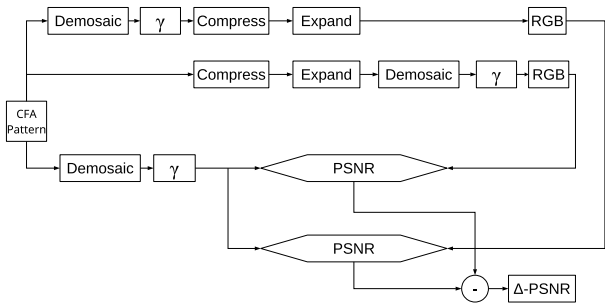

Fig. 8. Evaluation of the quality improvement for CFA compression: The image quality obtained by compressing the image in the gamma-corrected RGB domain is subtracted from the image quality obtained by compressing in the CFA domain.

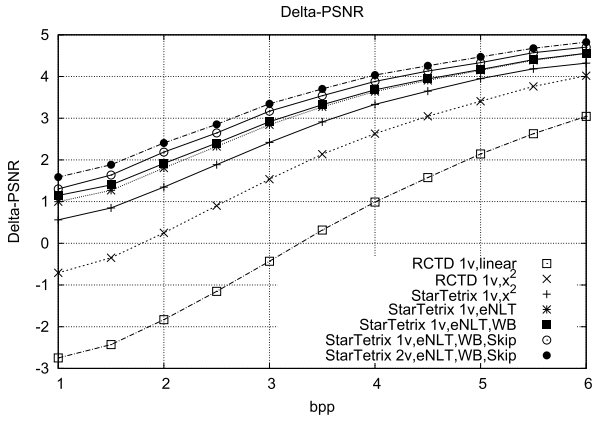

Fig. 9. PSNR of various configurations relative to compression in the RGB domain. Positive values indicate that compression of the CFA pattern is of advantage.

product of sensor gain factors and white balance is extracted from the camera file format, see section III for a discussion of these numbers. The resulting factors vary from image to image as the white balance depends on the scene, though the algorithm how the camera and its user made its selection is generally unknown. As the Star-Tetrix transformation accepts only powers of 2 for the white balance multipliers, the `dcraw` output is quantized to the nearest admissible number. The quantized values of the white balance exponents w_r and w_b lie for the test images in the set $\{0, 1, 2\}$.

What is reported in Fig. 9 is the average PSNR difference over the entire dataset of multiple CFA compression algorithms compared to compression in RGB domain. Positive values indicate an advantage of compression in the CFA domain, negative values a disadvantage. The plot "RCTD 1v,linear" shows the compression performance of the RCTD from eqn. 7 without pre- and post-emphasis and without white-balance correction, no non-linearity, using 1 vertical decomposition level and 5 horizontal wavelet decompositions.

As seen from the plot, however, it performs worse than compression in RGB space for bitrates lower than 3bpp, and only at bitrates higher than approximately 3bpp shows an advantage. Including a quadratic non-linearity already improves the performance significantly, as seen from the "RCTD 1v,x^2" plot. This method is, however, still outperformed by compression in the RGB domain at bitrates below approximately 2bpp. The

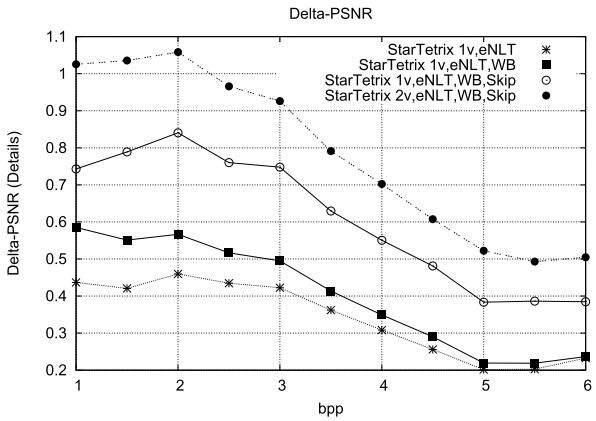

Fig. 10. PSNR improvements of various additional tools compared to the baseline configuration of the Star-Tetrix transformation using only the quadratic non-linearity.

Star-Tetrix transformation, however, (solid line, indicated by "Star Tetrix,1v,x^2") outperforms compression in the RGB domain for all bitrates. In the later plot, all four components undergo the wavelet transformation, the white-balance exponents are both 0 and the non-linearity is a simple square-root.

Replacing the simple quadratic nonlinearity by the extended non-linearity defined by eqn.(1) as indicated by the "StarTetrix 1v,NLT" plot, then adding white-balance correction ("StarTetrix 1v,NLT,WB"), skipping the wavelet decomposition ("StarTetrix 1v,NLT,WB,Skip") of the Δ component and finally adding a second wavelet decomposition level ("StarTetrix 2v,NLT,WB,Skip") outperforms the simple linear transformation by more than 4dB at the low end of tested bitrates. To make the PSNR improvements of these additional tools more visible, Fig. 10 shows only these improvements compared to the baseline Star-Tetrix transformation with a quadratic non-linearity, no white-balance and full wavelet decomposition with one vertical level. As seen there, the best possible configuration ("StarTetrix 2v,NLT,WB,Skip") provides an advantage of over 1dB below 2.5bpp compared to the baseline configuration of the same transform.

C. Preserving Values Below Optical Black

It also needs to be shown that it is important to preserve the values below the optical black, i.e. why $x < \Theta_1$ in eqn. (1) cannot be simply set to Θ_1. For this, we use night-shot images that contain a lot of noise along with pixel values below the optical black value. The compression algorithm is evaluated in a workflow that includes denoising, here by wavelet filtering of the CFA pixel values followed by deadzone quantization. The implementation is that of the open source `dcraw` algorithm, with the denoiser strength set to `-w 1500`; this value selects the relative size of the deadzone. The ground truth is here a dummy algorithm that clamps all values below the optical black value Θ_1 to Θ_1, where $\Theta_1 = 400$ for the given sources. Any compression algorithm that disregards values below the optical black cannot possibly perform better than this dummy. This (non-compression) algorithm is compared in Fig. 11 with a JPEG XS compression/decompression cycle

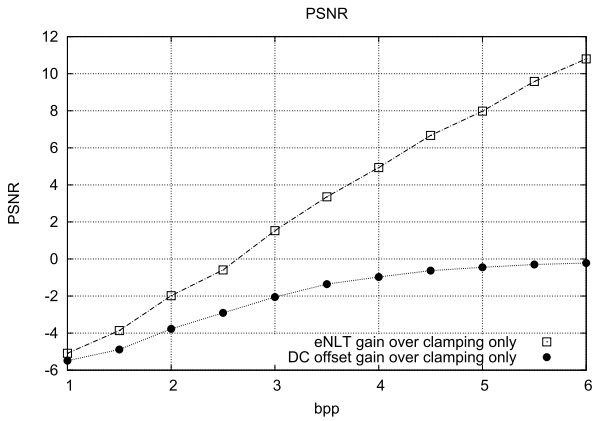

Fig. 11. Performance of two JPEG XS pre/post emphasis functions relative to a clamping-only workflow. Continuous: Clampling, square root pre-emphasis followed by compression. Dashed: Extended non-linearity.

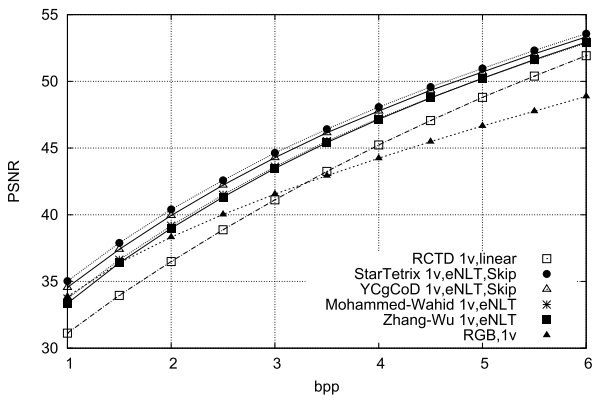

Fig. 12. PSNR of various decorrelation transformations, including compression in RGB space. All but the linear configuration include an extended non-linearity, but no white-balance correction, to allow a fair comparison.

which clamps to the optical black, and uses a square root pre-emphasis and squaring post-emphasis, and a JPEG XS compression/decompression cycle using the extended nonlinearity from eqn. (1). The output of all algorithms are converted into an RGB image by means of `dcraw` including the above denoising operation, and the PSNR between reconstructed images and an undisturbed, but otherwise identically denoised and demosaicked CFA pattern image is measured. Fig 11 plots PSNR *difference values* relative to the clamping-only workflow, i.e. the horizontal axis corresponds to the performance of pure clamping, but without any compression.

As seen in the figure, the pre-emphasis consisting of clamping and a square root is — as expected — bounded by the no-compression workflow that only disregards values below the optical black. This does not hold true for the extended non-linearity, which offers a PSNR improvement over 10dB for bitrates above 5.5bpp. Needless to say, this is a substantial gain which is due to the many dark pixels in both images.

D. Comparison With Other Decorrelation Filters

To compare our methods with the state of the art, i.e. [4], [6], [7], Fig. 12 shows absolute average PSNR values of the decorrelation transformations taken from the above works,

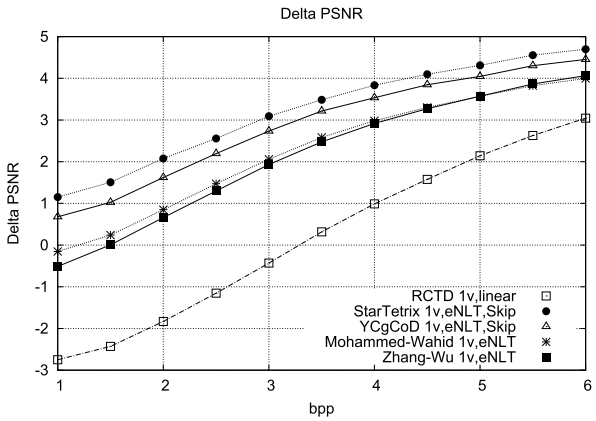

Fig. 13. Delta-PSNR of various configurations compared to compression in RGB space. The configurations are identical to that in Fig. 12.

including compression in the RGB domain. Entropy coding is all by JPEG XS. Since the differences between the configurations may be hard to spot, Fig. 13 shows the gain of the same configurations relative to the RGB baseline.

For fairness reasons, all transformations but compression in the RGB domain include the extended non-linearity pre-emphasis but no white-balance correction; the pre-emphasis is not discussed in the cited works, there transformations operate in linear light. White-balance correction as for the Star-Tetrix might be possible for [4], [6], [7], but would need to be implemented differently and thus would make the results hard to compare. For simplicity, we choose therefore to use unit white balance gains consistently in the experiment. Furthermore, we consistently use a use one vertical wavelet transformation only as this is the most-relevant configuration in practical applications.

The transformations discussed in [7] may include one or multiple iterations of decorrelation transformations, though then pile up additional latency. To configure it for a latency identical to Star-Tetrix, only a single iteration of the YCgCoD transformation is possible, and the transformation has to be rotated by 90 degrees as well, i.e. filtering in horizontal direction has to be exchanged by filtering in vertical direction and vice versa. This is of particular importance for the last filter step of the YCgCoD transformation which operates in vertical direction in [7], though becomes a filter in horizontal direction in our implementation.

In [5], authors discuss two possible decorrelation transformations, a simpler horizontal-only transformation, and a more extensive transformation that includes also a vertical transformation step, identical to the transformation in [6]. We choose to test only the latter, more extensive transformation as authors also report there that this is the better performing variant.

The transformation from [4] is identical to the first lifting step of eqn. (5); clearly, including additional steps improves the overall performance of the transformation. In [4], it is also observed that a related decorrelation transformation between CFA pattern channels can be obtained by an alternative (non-Mallat) wavelet decomposition. As this decomposition is not compatible with JPEG XS and also requires more than one vertical decomposition, we do not test it here.

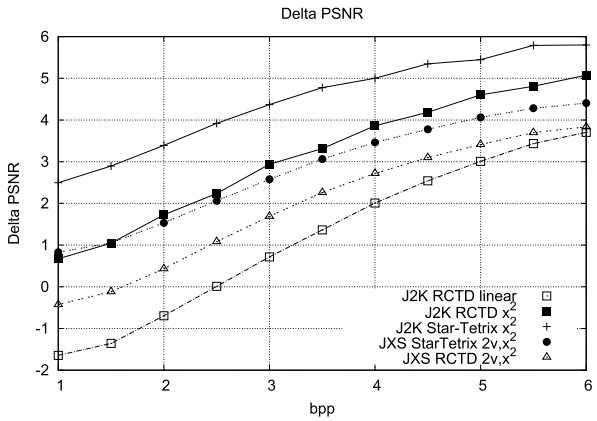

Fig. 14. PSNR of various JPEG 2000 and JPEG XS configurations relative to compression with JPEG XS in the RGB domain. Positive values indicate that compression of the CFA pattern is of advantage.

E. Other Entropy Coding Back-Ends

Next, we compare our proposal to that in [9]: the color decorrelation transformation discussed there is identical to the RCTD transformation, i.e. eqn. (7). In addition, the authors there also study the KLT transformation (RKLT in the paper) which is reported to perform better than the RCTD. However, as it is a data-dependent transformation, the entire image has to be available to derive it, and this is unsuitable for an online-compression algorithm such as JPEG XS.

Fig. 14 provides results of this and related JPEG 2000 configurations relative to compression of the RGB pattern. The plot denoted by "J2K RCTD linear" corresponds to the configuration selected in [9]. Here we use 5 decomposition levels, the lossy 5/3 filter, 3 guard bits — one additional guard bit to address potential overflows due to the RCTD transformation — 1 layer and scalar derived quantization of 19 bits. Pre-processing with the RCTD and/or the Star-Tetrix transformation and the non-linearity is performed outside of JPEG 2000 with 20 bits similar to JPEG XS and then scaled to a nominal bit-depth of 14 bits, allowing one additional sign bit and one guard bit to allow for overflows in the preprocessing step. Preprocessing outside of the encoder is currently necessary as JPEG 2000 does not specify the Star-Tetrix transformation, and we use preprocessing consistently with all configurations.

Note that for this first configuration considered, no pre-emphasis is used prior compression; in [9] a decoder-side non-linearity reproduced by an ICC profile converts from linear light to a gamma-corrected color space. In our setup, this non-linearity is part of the CFA to RGB conversion process implemented by `dcraw`. As seen in the plot, the "J2K RCTD linear" configuration leads to non-ideal performance compared to compression of the CFA pattern in RGB space, but readers should be aware that the main focus of [9] is to provide a configuration of JPEG 2000 that allows both CFA pattern compression, and reconstruction of the same codestream to an RGB image where the conversion to the RGB image is performed by a (differently configured) JPEG 2000 reconstruction process.

If this design constraint is relaxed, and a quadratic pre- and post-emphasis function are included, and the 5/3 filter is replaced by the 9/7 filter, performance is improved considerably and we arrive at the plot indicated by "J2K RCTD x^2". This configuration is relevant because it can be realized by means of JPEG 2000 part 2 [25]: The square non-linearity can be represented by an NLT marker, and so can the RCTD transformation by an appropriate multi-component decorrelation (MCT) marker, see again [9]. Despite having 5 vertical decomposition levels instead of the maximal 2 allowed by JPEG XS, and a much more elaborated entropy coding back-end, namely EBCOT [23], the performance at bitrates below 4bpp is comparable or only slightly better than that of JPEG XS using the Star-Tetrix transformation. If, in addition, we combine Star-Tetrix with JPEG 2000, we arrive at a PSNR gain of about 2.5dB over a similar JPEG XS configuration with only 2 vertical levels, see the plot J2K-Star-Tetrix x^2. This configuration is not covered by the JPEG 2000 standard at this time. Note, however, that this comes with additional complexity, see Fig. 7.

VIII. Discussion

As seen in Fig. 9, compressing in the CFA domain may provide a performance advantage in terms of quality in the RGB domain for the right choice of coding tools. Intuitively, an image compression algorithm operating in the CFA pattern domain only requires to compress one third of the data of an algorithm operating on RGB data for the same image size, and one might expect that it should always perform better.

The reader should, however, keep in mind that the additional sample values of the image in the RGB domain are made up by the demosaicking algorithm and are therefore redundant, so the above hand-waving argument cannot hold true in generality. As seen in the figure, it may be correct for high bitrates where signal noise becomes dominant. Also note that not all of the redundancy generated by the demosaicking algorithm can be effectively removed by a linear decorrelation filter such as the wavelet in the JPEG XS compressor, especially keeping in mind that the AHD algorithm used in the evaluation here contains non-linearities due to its data-dependent choice of the filter direction. At low bitrates, however, there is less redundancy in the CFA pattern than in the RGB data, such that compressing the CFA pattern is harder than the RGB image.

If we understand "CFA compression" in a way that measures quality of the demosaicked image in the RGB domain, then including coding tools that mimic similar tools in the demosaicking pipeline bring a significant advantage in compression performance. That is, demosaicking accesses neighbouring pixels to interpolate channels at all pattern positions, and thus the transformation includes inter-component and spatial filtering. Conversion to RGB typically implies a conversion to a gamma-corrected color space, and thus including an approximation of this non-linearity in the compression workflow provides a significant advantage. Modifying the transformation to mirror white-balance corrections typically applied as part of the demosaicking algorithm further improves the transformation.

The additional tools proposed in this work are not *exact* lookalikes of those applied in the demosaicking algorithm as JPEG XS is constrained in terms of complexity and latency. That is, we replaced the 2.2 exponent of the sRGB curve by squaring, i.e. a multiplication, to avoid large lookup tables or complex functions. Square-root algorithms of the complexity of a division are well-known, e.g. the Toepler algorithm or variants of the CORDIC algorithm [28] and allow as well simple encoder implementations. We also restrict the Star-Tetrix transformation to access only nearest-neighbours during filtering to limit its overall latency. A looser latency constraint allows better decorrelation, e.g. by either including a second vertical wavelet decomposition, or — for the YCgCo transformation — additional iterations.

It should, however, be noted that not all tools discussed here are useful in all applications: The non-linearity creates by its construction a loss and thus must be omitted if lossless coding is required. As the white-balance correction is implemented through a modified lifting step, it can be carried over to the lossless case, though. The Star-Tetrix transformation is by construction lossless and thus can be applied as-is in the lossless case as well.

Some workflows may not require the non-linearity, or may require a modified version of it. In industrial applications, the camera output is often in a non-linear (e.g. logarithmic) space to cover a large range of illumination conditions, and thus an additional non-linearity in the compression pipeline is of little value, or should be adapted to this particular use case. Also, the non-linearity introduced here only approximates the gamma correction typical in SDR workflows, and is likely less suitable if additional grading operations typical for HDR processing become part of the overall process. The JPEG committee is currently looking into such extensions of the standard in a planned 3rd edition of the specification.

ACKNOWLEDGMENT

The authors would like to thank Nikon for providing their test data set for the experiments carried out in this work.

REFERENCES

[1] E. B. Bayer, "Color imaging array," US Patent Office, U.S. 3 971 065, Jul. 20, 1976.

[2] *AIA Global Vision Systems Trade Association, Vision Standards*. Accessed: Aug. 6, 2020. [Online]. Available: https://www.visiononline.org/vision-standards.cfm

[3] *Japan Industrial Imaging Association (JIIA)*. CoaXPress. Accessed: Aug. 6, 2020. [Online]. Available: http://jiia.org/en/standard_dl

[4] N. Zhang and X. Wu, "Lossless compression of color mosaic images," in *Proc. Int. Conf. Image Process. (ICIP)*, Singapore, vol. 1, 2004, pp. 517–520, doi: 10.1109/ICIP.2004.1418804.

[5] S. K. Mohammed and K. A. Wahid, "Lossless and reversible colour space transformation for Bayer colour filter array images," *IET Image Process.*, vol. 12, no. 8, pp. 1485–1490, Aug. 2018.

[6] S. K. Mohammed, K. M. M. Rahman, and K. A. Wahid, "Lossless compression in Bayer color filter array for capsule endoscopy," *IEEE Access*, vol. 5, pp. 13823–13834, 2017, doi: 10.1109/ACCESS.2017.2726997.

[7] T. Suzuki, "Lossless compression of CFA-sampled images using YDG-COCG transforms with CDF wavelets," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Athens, GA, USA, Oct. 2018, pp. 1143–1147.

[8] T. Suzuki, "Wavelet-based spectral–spatial transforms for CFA-sampled raw camera image compression," *IEEE Trans. Image Process.*, vol. 29, pp. 433–444, 2020.

[9] 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, doi: 10.1109/TMM.2017.2741426.

[10] K. Hirakawa and T. W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 360–369, Mar. 2005, doi: 10.1109/TIP.2004.838691.

[11] H. Malvar, "System and method for encoding mosaiced image data employing a reversible color transform," U.S. Patent Office 7 480 417, Jan. 20, 2006.

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

[13] 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, doi: 10.1109/TCSVT.2014.2302546.

[14] 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, Feb. 2008, doi: 10.1109/TIP.2007.914153.

[15] T. Richter, J. Keinert, A. Descampe, and G. Rouvroy, "Entropy coding and entropy coding improvements of JPEG XS," in *Proc. Data Compress. Conf.*, Mar. 2018, pp. 87–96.

[16] T. Richter, J. Keinert, S. Fößel, A. Descampe, G. Rouvroy, and J. B. Lorent, "JPEG-XS—A high quality mezzanine image codec for video over IP," *SMPTE Motion Imag. J.*, vol. 127, no. 9, pp. 39–49, Oct. 2018.

[17] A. Descampe, J. Keinert, T. Richter, S. Fößel, and G. Rouvroy, "JPEG XS, a new standard for visually lossless low-latency lightweight image compression," *Appl. Digit. Image*, vol. 10396, Sep. 2017, Art. no. 103960M.

[18] S. H. Park, H. S. Kim, S. Lansel, M. Parmar, and B. A. Wandell, "A case for denoising before demosaicking color filter array data," in *Proc. Conf. Rec. 43rd Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, 2009, pp. 860–864.

[19] H. Akiyama, M. Tanaka, and M. Okutomi, "Pseudo four-channel image denoising for noisy CFA raw data," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Quebec City, QC, Canada, Sep. 2015, pp. 4778–4782.

[20] *Electronic Still-Picture Imaging—Removable Memory—Part2: TIFF/EP Image Data Format*, Standard ISO/IEC 12234-2, 2001.

[21] *Multimedia Systems and Equipment—Colour Measurement and Management—Part 2-1: Colour Management—Default RGB Colour Space—sRGB*, document IEC as IEC 61966-2-1, 1999.

[22] *Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange*, document ITU-R Rec. BT.2020-2, 2015.

[23] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[24] M. Boliek Ed., *Information Technology—JPEG 2000 Image Coding System*, Standard ISO/IEC 15444-1, 2000.

[25] M. Boliek Ed., *Information Technology—JPEG 2000 Image Coding System: Extensions—Part 2*, Standard ISO/IEC 15444-2, 2004.

[26] D. Taubman and M. Marcellin, *JPEG2000: Image Compression, Fundamentals, Standards and Practise*. Norwell, MA, USA: Kluwer, 2002.

[27] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[28] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.

[29] D. S. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practise* (Series in Engineering and Computer Science). Boston, MA, USA: Springer, 2002.

[30] A. N. Netravali and B. G. Haskel, *Digital Pictures: Representation and Compression*. New York, NY, USA: Plenum, 1988.

[31] T. Richter, "Rate allocation for bayer-pattern image compression with JPEG XS," in *Proc. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Mar. 2019, pp. 320–328, doi: 10.1109/DCC.2019.00040.

[32] T. Richter and S. Fößel, "Bayer pattern compression with JPEG XS," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 3177–3181, doi: 10.1109/ICIP.2019.8803376.

Thomas Richter (Member, IEEE) received the Ph.D. degree in mathematical physics from the Berlin Institute of Technology, TU Berlin, Berlin, Germany, in 2000.

He moved to AllgoVison Technology GmbH for working on Joint Photographic Experts Group (JPEG) 2000. In 2002, he moved to the Multimedia Center, TU Berlin. In 2007, he moved to the Computing Center, University of Stuttgart, Stuttgart, Germany. Since 2017, he has been with Fraunhofer IIS, Erlangen, Germany, where he is currently a Senior Research Scientist. Since 2001, he has been a member of ISO SC29 WG01 (JPEG). He edited multiple standards, among them ISO/International Electrotechnical Commission (IEC) 18477 (JPEG XT) and ISO/IEC 21122-1 (JPEG XS).

Siegfried Fößel (Senior Member, IEEE) received the Diploma and Ph.D. degrees in electrical engineering from Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, in 1989 and 2000, respectively.

He is currently the Head of the Moving Picture Technologies Department, Fraunhofer IIS, Erlangen. He is a Professor of the Technology Study Program with the HFF Film School, Munich, Germany. He is a spokesperson of the Fraunhofer Business Area Digital Media. He has published more than 70 publications. His previous research projects include the development of digital cinema cameras, such as ARRI D20/D21, certification test plan for digital cinema initiatives (DCI), standardization of digital cinema profiles for the Joint Photographic Experts Group (JPEG) 2000 and development of postproduction tools for digital cinema (easyDCP and IMF), and immersive audio (MPEG-H). He is an SMPTE fellow. He is a member and the chair of various national and international standardization bodies. He serves as the President of the Fernseh-und Kinotechnische Gesellschaft (FKTG), the German equivalent to SMPTE.

Antonin Descampe received the master's degree in telecommunications from CentraleSupelec (formerly Ecole Centrale Paris), Gif-sur-Yvette, France, in 2002, and the master's and Ph.D. degrees in electrical engineering from the Université Catholique de Louvain (UCLouvain), Ottignies-Louvain-La-Neuve, Belgium, in 2003 and 2008, respectively. During his Ph.D. and postdoctoral research, he worked in the field of machine learning and image processing. In this context, he cofounded intoPIX, Mont-Saint-Guibert, Belgium, where he has been involved in the development and standardization of new image and video compression technologies. He is currently an Assistant Professor with the Media Innovation and Intelligibility Laboratory (MiiL), UCLouvain. He has been one of the project leader of the Joint Photographic Experts Group (JPEG) XS standardization effort. His research interests include image and video processing, algorithmic accountability, and AI in the media and journalistic context. He is an Active Member of ISO (JPEG and MPEG), VSF, and SMPTE.

Gaël Rouvroy received the master's and Ph.D. degrees in electronics and mechanics engineering from the Université Catholique de Louvain, Ottignies-Louvain-La-Neuve, Belgium, in 2001 and 2004, respectively. He is currently a leading expert in electronic design and image algorithms. He has presented more than 30 articles on security, watermarking, image compression, and design strategies for cost-effective and low-power implementations. He is the inventor of several patents. He is a Co-founder of intoPIX, Mont-Saint-Guibert, Belgium, where he is currently the Chief Executive Officer and the Director of technology. His specializations are secure and reconfigurable hardware compression, cryptographic hardware, block ciphers, DRM, watermarking, and image and video compression. He is active in several standardization committees.