H.264-Based Compression of Bayer Pattern Video Sequences

Colin Doutre, *Student Member, IEEE*, Panos Nasiopoulos, *Member, IEEE*, and
Konstantinos N. Plataniotis, *Senior Member, IEEE*

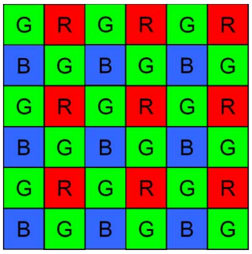
***Abstract*—Most consumer digital cameras use a single light sensor which captures color information using a color filter array (CFA). This produces a mosaic image, where each pixel location contains a sample of only one of three colors, either red, green or blue. The two missing colors at each pixel location must be interpolated from the surrounding samples in a process called demosaicking. The conventional approach to compressing video captured with these devices is to first perform demosaicking and then compress the resulting full-color video using standard methods. In this paper two methods for compressing CFA video prior to demosaicking are proposed. In our first method, the CFA video is directly compressed with the H.264 video coding standard in 4:2:2 sampling mode. Our second method uses a modified version of H.264, where motion compensation is altered to take advantage of the properties of CFA data. Simulations show both proposed methods give better compression efficiency than the demosaick-first approach at high bit rates, and thus are suitable for applications, such as digital camcorders, where high quality video is required.**

***Index Terms*—Bayer pattern, color demosaicking, H264/AVC, single-sensor digital video cameras, video coding.**

I. Introduction

MOST commercial digital cameras use a single light sensor which is monochromatic in nature. In order to capture RGB color information, a color filter array (CFA) is used, which produces a mosaic image where each pixel location contains either a red, green or blue sample. The Bayer pattern CFA [1] is commonly used, which captures pixels in groups of four; each group containing two green, one red and one blue sample (Fig. 1). More green samples are captured than red or blue because the human visual system is more sensitive to the green portion of the light spectrum. Other CFA patterns are possible [2] but the Bayer pattern is considered here due to its commercial importance.

In order to form a full color image or video from CFA data, the color data is interpolated in a process called demosaicking [3]–[9]. The process of demosaicking still images has been extensively studied. For a comprehensive literate review we refer the reader to [3]. Advanced demosaicking methods use techniques such as edge detection and spectral models to reduce false-color artifacts that are common in interpolated CFA images.



G	R	G	R	G	R
B	G	B	G	B	G
G	R	G	R	G	R
B	G	B	G	B	G
G	R	G	R	G	R
B	G	B	G	B	G

Fig. 1. Bayer pattern CFA.

When video is captured with a CFA, demosaicking is usually done on each frame independently. However recent work has explored using motion estimation (ME) to improve performance when demosaicking is performed on a video [10], [11]. Through ME, captured samples from other frames can be used to enhance the estimates for missing pixels in each frame; i.e., temporal correlation between frames is exploited in the demosaicking process. In [10], the final value for each missing pixel is obtained by combining an estimate obtained through conventional single frame demosaicking with an estimate obtained through ME. In [11], a video demosaicking method is proposed where multiple estimates for each missing pixel are obtained through ME, each estimate being obtained from a different frame. These estimates are adaptively combined to generate the final value for each missing pixel.

Another important processing stage in digital cameras is data compression, where redundancies are removed from the data in order to reduce storage requirements. Compression techniques can be classified as either lossy or lossless. In lossless compression, the original data is restored exactly after decompression. Lossy compression achieves much greater compression ratios by allowing some amount of distortion in the decompressed data. The main goal of lossy compression is to maximize the quality achieved at a given bit rate, or equivalently, minimize the bit rate needed to achieve a given level of quality.

Still image compression removes spatial redundancies within an image, often by taking the discrete cosine transform (DCT) of blocks of pixels and coding the transform coefficients. Video compression attempts to remove redundancies both within a single frame and between frames. A key technique used in major video coding standards is motion compensation (MC). When

Manuscript received September 18, 2006, revised November 21, 2007. This work was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). This paper was recommended by Associate Editor O. C. Au.

C. Doutre and P. Nasiopoulos are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: colind@ece.ubc.ca; panos@ece.ubc.ca).

K. N. Plataniotis is with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: kostas@comm.utoronto.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2008.919111

1051-8215/$25.00 © 2008 IEEE

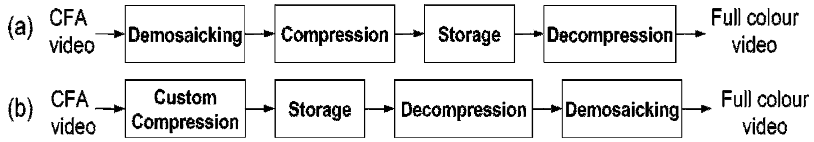


Fig. 2. Flow diagram of CFA compression schemes. (a) Traditional demosaick- first approach. (b) Method with compression prior to demosaicking.

MC is used, some frames are coded by predicting the frame from previously coded frames, and storing the difference image between the actual frame and the prediction. Such frames are called P-frames. MC involves dividing a frame into blocks and estimating and coding a displacement vector for each block. The displacement vector tells the decoder which block in previously coded reference frame to use as a prediction. Using the displacement vector the decoder can form the same prediction for the block, and by adding the difference image to the prediction the original frame can be restored.

The conventional approach to compressing CFA data (still image or video) is to first perform demosaicking and then compress the resulting full color data [Fig. 2(a)]. This approach is sub-optimal because the amount of data is expanded by a factor of three in the demosaicking stage, which increases the compression processing time and introduces redundancy that the compression stage must remove. To avoid these problems, compression can be carried out on the CFA data prior to demosaicking [Fig. 2(b)] [12]–[19].

A few papers have been written on lossy compression of CFA still image data, with the goal being to provide better quality at the same bit rate than the conventional demosaick-first approach, allowing the camera to either store higher quality images, or reduce storage requirements while maintaining the same quality. In [12], Lee and Ortega propose a method starting with a modified YCbCr color space conversion performed on each group of four Bayer pattern samples. This produces a luma sample at each green pixel location and a chroma sample at each red or blue location. The Cb and Cr samples are put into separate arrays and compressed with JPEG. The luma samples are rotated 45° to form a rhombus shape, and then compressed with a modified JPEG algorithm.

In [13], Koh *et al.* propose two techniques for compressing CFA still image data. Both consist of an YCbCr color space conversion (as in [12]), followed by forming rectangular arrays out of the luma and chroma data which are then compressed with JPEG. After the YCbCr conversion, the luma samples are arranged in the shape of the green samples in the Bayer pattern [Fig. 3(a)]. The structure conversion method consists of merging every two columns of luma samples into a single column [Fig. 3(b)]. The structure separation method involves separating the even column and odd column luma samples into two distinct rectangular arrays [Fig. 3(c)]. The structure conversion method requires altering the relative positioning of the even and odd column samples, which distorts the image slightly. The structure separation method avoids the image distortion problem by downsampling the even column and odd column samples into separate arrays; however, it does not exploit the correlation between the two resulting arrays.

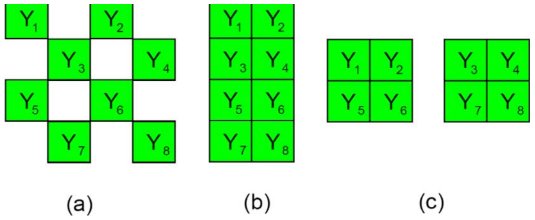


Fig. 3. Pixel rearranging methods in [13]. (a) Original arrangment of luma samples. (b) Structure conversion. (c) Structure separation.

Other lossy compression methods for CFA still images involve techniques such as sub-band coding [14], and vector quantization of groups of CFA pixels [15].

A number of techniques for lossless compression of CFA still image have been proposed. These include using structure arrangements proposed in [13] followed by compression with JPEG-LS [16], and an integer wavelet transform method [17]. A real-time system for lossless compression of CFA video is presented in [18], which uses the structure separation method from [13] together with simple MC and entropy coding.

The only work that addresses the issue of lossy compression of CFA video data is presented in [19]. In that paper, the Bayer pattern data is split into three separate arrays of green, red and blue pixels using the structure conversion method of [13]. These three arrays are compressed with a custom MPEG-2 like video coding scheme. A custom video coding scheme was developed because no major video coding can compress data where one color channel (green) has twice the vertical size and the same horizontal size as the other two colors (red and blue). The motion vectors from the green channel are used on the red and blue channels (with appropriate downsampling and scaling), to avoid the need for calculating and encoding three sets of motion vectors. An IBBPBBPBBPBB group of pictures structure is used, with JPEG compression used to compress the I-frames and residual for P– and B-frames. This method has poor P-frame and B-frame performance because CFA data generally contains severe aliasing, resulting in reduced temporal correlation (aliasing in CFA data is discussed in Section II of this paper).

In this paper two new methods are proposed for compressing CFA video data, one using the H.264 video coding standard and one using a modified version of the H.264. H.264 is the latest major video coding standard and it provides significant improvements in coding efficiency over previous standards such as MPEG-2. Basing our methods on H.264 allows us to exploit the latest powerful video coding tools. Our first proposed method compresses the CFA video with standard H.264 and achieves better quality (measured with mean-square error) than the demosaick-first approach at high bit rates. Our second

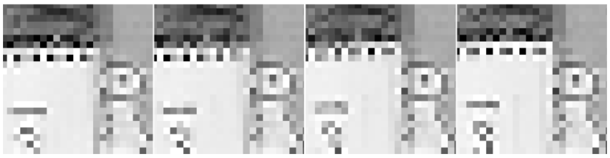


Fig. 4. Four successive frames of red Bayer pattern data, illustrating the affect of aliasing and the low correlation between frames.

method further increases compression efficiency by introducing a modified MC scheme into H.264, alleviating problems that arise due to aliasing in the CFA data. Both methods are suitable for devices such as digital camcorders where video is encoded with high quality.

The rest of the paper is organized as follows. In Section II, aliasing in CFA data and its negative effect on video coding is discussed, providing the motivation for a key part of the proposed method. The proposed methods for compressing CFA video data are described in Section III. Simulation results showing the performance our methods relative to the conventional demosaick-first approach are presented in Section IV, along with a comparison of the complexity the different approaches. Conclusions are presented in Section V.

II. Impact of Aliasing on CFA Video Coding

Aliasing in video has been shown to negatively impact the coding of P-frames [20]. If there is movement between frames the effect of aliasing will be different in each frame. This results in lower correlation between frames, and hence large P-frame size. The negative effect of aliasing can be reduced by using sub-pel accurate motion vectors together with adaptive interpolation filters [21].

CFA data can contain severe aliasing [22]. Single sensor cameras usually use an optical filter to limit aliasing [23]. When selecting how much filtering to use, there is a tradeoff between limiting aliasing and capturing fine image detail. Furthermore, since the Bayer pattern contains more green samples than red or blue, different amounts of filtering are needed to limit aliasing in the different colors. If enough filtering were used so that there was little aliasing in the red and blue channels, then significant detail would be lost from the green channel which is undesirable, and defeats the purpose of sampling green at a higher rate than red or blue.

An assumption sometimes made in demosaicking research is that enough optical filtering is done so that if a full color image were captured, it would contain negligible aliasing, however sampling with the Bayer CFA introduces aliasing [24]. Other work makes the assumption that significant aliasing occurs in the red and blue channels, but not in the green [5], [8].

The effect of aliasing in CFA video is illustrated in Fig. 4, which shows a portion of four successive frames of red Bayer pattern data. The calendar that covers the left portion of each frame is moving vertically. Notice how the moving portion, especially the number “3”, looks considerably different in each frame due to aliasing.

Advanced demosaicking algorithms reduce the effects of aliasing in each color channel by using information from the other color channels [5], [8], [10], [11]. An example of this is shown in Fig. 5, which presents the four frames of red data in Fig. 4 after demosaicking has been performed with the method presented in [9]. Demosaicking increases the amount of red data by a factor of four, so the frames in Fig. 5 are bigger than in Fig. 4. We observe that there is considerably more temporal correlation in the frames after demosaicking than there is in the original CFA data. Since each CFA frame is a subset of the corresponding demosaicked frame, the CFA frames can be effectively predicted from the demosaicked versions of the other frames.

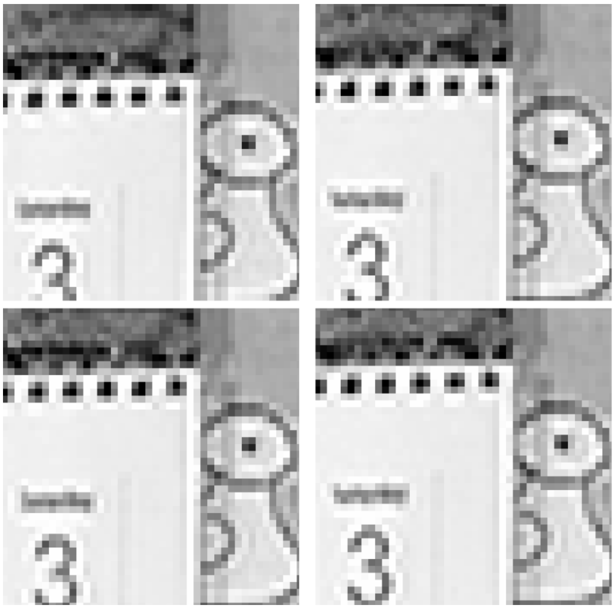


Fig. 5. Four frames of red data in Fig. 4 after demosaicking.

III. Proposed Methods

Both of our proposed methods involve dividing the CFA data into separate arrays of green, blue and red data, which are compressed in 4:2:2 sampling mode. In our first method, standard H.264 is used for compressing the arrays of red, green and blue. In our second method, a modified MC scheme is also applied, where demosaicking is performed on the reference frames within the encoder and decoder to reduce the negative effects of aliasing on P-frames. The method of creating rectangular arrays of each color and arranging the data for compression with H.264 is described in Section III-A. The modified MC scheme used in our second method is described in Section III-B.

A. Pixel Rearranging

Most video compression standards, including H.264, can only compress video of rectangular shape, so in order to compress the CFA data using H.264, the pixels must be rearranged into rectangular arrays. The red and blue data are sampled in a rectangular manner, so they can easily be separated into arrays one quarter the size of the Bayer data.

In [13], two options for creating rectangular arrays of quincunx sampled data are proposed (Fig. 3). In [13], the methods are applied to luma samples after a color space conversion; here we apply them to the green samples directly without a color space conversion. If a frame of Bayer pattern data has dimensions $M \times N$, the structure separation method involves separating the green data into two arrays of size $(M/2) \times (N/2)$, one containing the even column samples and the other containing the odd column samples [Fig. 3(c)]. Implementing this method in a video codec would require extensive modification to allow four channels to be compressed together rather than the usual three. Also, downsampling the green data into two separate channels introduces further aliasing in each channel (in addition to the aliasing introduced due to Bayer sampling the green channel). In [13], a lowpass filter is applied to the green data before downsampling to limit the aliasing. However, applying lowpass filtering is undesirable since it removes high-frequency detail which could be used when demosaicking is later applied to the data.

The structure conversion method in [13] involves merging the two green samples from every group of four Bayer pattern samples into a single column, resulting in a green channel of size $M \times (N/2)$. This method does not suffer from the aliasing problems of the structure separation method, however the merging process does distort the data somewhat [Fig. 3(b)]. In [19], the structure conversion method is used for compressing video, where they compress a green channel of size $M \times (N/2)$ together with red and blue channels of size $(M/2) \times (N/2)$. Since the relative dimensions of the three color channels do not correspond to any sampling scheme supported in major video coding standards, a custom codec was used, where the motion vectors from the green channel are reused on the red and blue channels, with appropriate downsampling and scaling on the motion vectors.

We proposed a different structure conversion method, where the samples are merged into rows (Fig. 6). Let $f(i,j)$ be the value of the CFA data at spatial location (i,j) within the image, where i donotes the row and j the column. Let $R(i,j)$, $G(i,j)$ and $B(i,j)$ denote the arrays of red, green and blue data after conversion to separate arrays. The color arrays can be expressed in terms of the CFA data by the following equations:

$$\begin{aligned} G(i,j) &= \begin{cases} f(2i,j), & j \text{ even} \\ f(2i+1,j), & j \text{ odd} \end{cases} \\ B(i,j) &= f(2i+1,2j) \\ R(i,j) &= f(2i,2j+1). \end{aligned} \tag{1}$$

The array $G(i,j)$ has dimensions $(M/2) \times N$, $B(i,j)$ and $R(i,j)$ have dimensions $(M/2) \times (N/2)$. This approach allows the data to be compressed in 4:2:2 sampling mode, with green data in the luma channel and red and blue data in the chroma channels. Since 4:2:2 sampling support was added to H.264 with the Fidelity Range Extensions (FRExt), the CFA data can be compressed with H.264 achieving the same effect as in [19] (reusing motion vectors from the green channel on the red and blue data) without the need for a custom codec. Our first proposed method simply consists of compressing the green, blue and red arrays given by (1) with standard H.264.

B. Modified Motion Compensation

MC is the key technique that distinguishes video compression from still image compression. In general, exploiting temporal correlation through ME/compensation is what distinguishes video processing from still image processing. This is true for both full colour data and CFA data; for example the CFA video demosaicking methods in [10] and [11] employ ME to improve the quality of the demosaicked video. To improve the performance of our CFA video compression method, we modify the way ME and MC is done based on the properties of CFA data.

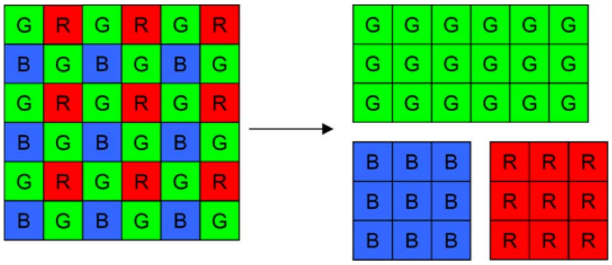


Fig. 6. Conversion from mosaic data into separate R, G, and B arrays.

As discussed in Section II, the aliasing introduced when sampling with a CFA negatively effects P-frame coding. Also, subpixel interpolation is a key factor in MC based video coding, which is used in all recent video coding standards including MPEG-2, H.263, and H.264. In all existing video coding standards when a reference frame is interpolated to allow sub-pixel accurate MC, only information from the current colour channel is used. In CFA data, the colour channels are sampled at different spatial locations, and there is a high amount of correlation between the colour channels (spectral correlation). Consequently, information from the other two colour channels can be used to increase the accuracy of interpolation during the MC process.

In our second proposed method we minimize the problem of aliasing in CFA data by performing demosaicking on the reference frames in the encoder and decoder for the purposes of MC. Each P-frame of CFA data is predicted from the demosaicked reference frames. The demosaicking process exploits correlation between the colour channels, resulting in more accurate interpolation between captured samples. This provides a better prediction for the frame being coded and hence lowers the bit rate.

In H.264, I- and P-frames are used for predicting other frames of the video. After a frame has been encoded, it is decoded within the encoder, and the decoded version of the frame is used for prediction. In our method rather than directly using the decoded frame for prediction, demosaicking is first performed on the decoded frame, and the demosaicked frame is used for prediction. This increases the size of the green data by a factor of two and the red and blue data by a factor of four. This is illustrated in Fig. 7(a), which shows a block of pixels from a decoded frame, and the block after demosaicking. The numbers in Fig. 7(a) indicate the location of each of the original pixels in the demosaicked frame and the unlabeled pixels are generated in the demosaicking process. Any demosaicking method could be used on the reference frames. The choice of a particular method would depend on the application and the amount of complexity that can be tolerated. Different demosaicking methods are evaluated for this purpose in Section IV-A.

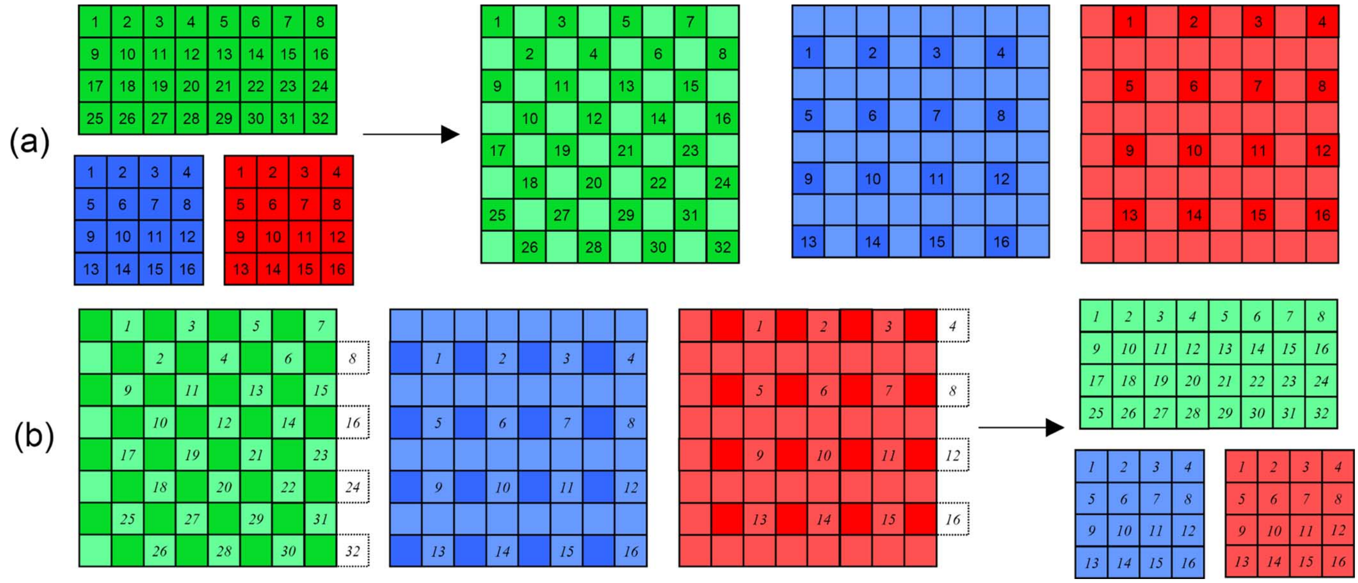


Fig. 7. (a) Performing demosaicking on a block of pixels from a reference frame. (b) Sampling the demosaicked reference frame to obtain a prediction for the block when the motion vector is (1,0) in full pel units.

After demosaicking, all three color channels are up-sampled by a factor of four in the horizontal and vertical directions to support quarter pel accurate motion vectors. This is done using the method defined in the H.264 standard for upsampling the luminance channel (where a 6-tap FIR filter generates the half-pel samples). RGB data has properties more similar to luma than chroma, so the luma upsampling method is used to give better performance.

In the H.264 reference encoder, ME is done on the luma channel. When ME is performed on RGB data, the green channel is usually used [10], [11], since the green data is more high correlated with luma than red or blue. So in our second proposed method, ME is done on the green channel. Consider a green pixel at location (i_G, j_G) in a CFA frame. After demosaicking, this pixel will be located at position $(2i_G, j_G)$ in the demosaicked frame if j_G is even and position $(2i_G + 1, j_G)$ if j_G is odd (Fig. 7(a)). Let Ω denote the set of coordinates of the green pixels within a block in the CFA data. A motion vector (m_i, m_j) is calculated for the block by minimizing the sum of absolute differences (SAD) given by

$$\mathrm{SAD} = \sum_{(i,j)\in\Omega} \begin{cases} G(i,j) - G_{\mathrm{dem}}(2i + m_i, j + m_j), & j \text{ even} \\ G(i,j) - G_{\mathrm{dem}}(2i + 1 + m_i, j + m_j), & j \text{ odd} \end{cases} \tag{2}$$

where $G_{\mathrm{dem}}(i,j)$ is the demosaicked reference frame being used for prediction. Note that in (2), the demosaicked reference frame is being sampled with shape of the green data in the Bayer pattern, with the motion vector controlling the relative position of the sampling. After a full pel motion vector has been found with (2), the motion vector is refined to quarter pel accuracy, as is done in the H.264 reference encoder.

In our method, the motion vectors calculated from the green channel are also used on the red and blue channels. A red pixel at location (i_R, j_R) will be moved to $(2i_R, 2j_R + 1)$ after demosaicking, and a blue pixel at (i_B, j_B) will be moved to $(2i_B + 1, 2j_B)$. In order to obtain a prediction for a pixel in a CFA frame using MC, the motion vector calculated is added to the corresponding position of the pixel in the demosaicked frame, and the demosaicked frame is sampled at that position. Let $B_{\mathrm{dem}}(i,j)$, and $R_{\mathrm{dem}}(i,j)$ be the values of the demosaicked frame being used for prediction, and the motion vector for a block of CFA data be (m_i, m_j). Then the predictions for the CFA pixels in the block will be:

$$\begin{aligned} &G_{\mathrm{pred}}(i_G, j_G) \\ &\quad = \begin{cases} G_{\mathrm{dem}}(2i_G + m_i, j_G + m_j), & j_G \text{ even} \\ G_{\mathrm{dem}}(2i_G + 1 + m_i, j_G + m_j), & j_G \text{ odd} \end{cases} \\ &B_{\mathrm{pred}}(i_B, j_B) \\ &\quad = B_{\mathrm{dem}}(2i_B + m_i + 1, 2j_B + m_j) \\ &R_{\mathrm{pred}}(i_R, j_R) \\ &\quad = R_{\mathrm{dem}}(2i_R + m_i, 2j_R + 1 + m_j). \end{aligned} \tag{3}$$

As an example, consider the task of predicting the 8 × 4 block of CFA pixels in Fig. 7(a) in a future frame when the motion vector is (1,0) in full pel units. Fig. 7(b) shows how the demosaicked frame is sampled to obtain a prediction for the block. The white squares represent pixels that are obtained by edge replication.

In summary, our MC scheme uses demosaicked versions of reference frames to predict the CFA frame being coded. This takes advantage of the increased temporal correlation of frames after demosaicking has been performed, without the need for compressing the larger demosaicked frames themselves.

IV. Results

To evaluate the performance of our compression scheme two videos from the SVT High Definition Test Set [25], *CrowdRun* and *OldTownCross*, were used in our simulations (Fig. 8). The

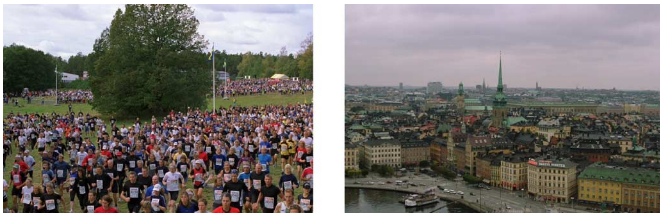


Fig. 8. Screenshots of the test videos, CrowdRun (left), and OldTownCross (right).

CrowdRun sequence is a shot of hundreds of marathoners running through a park. It contains a high amount of motion due to the runners and also slight camera motion. The *OldTownCross* sequence is a aerial shot of a European city containing camera motion. Both videos contain large amounts of fine detail (high frequency image content).

These videos were digitized at a resolution of 3840 × 2160, 16 bits per (RGB) color plane, 50 fps. We downsampled and cropped the videos to give 720 × 480, 8 bit RGB data at 25 fps. This is the quality of video typically captured by consumer digital camcorders, our target application. CFA videos were obtained by sampling the RGB videos with the Bayer pattern.

In all tests, 60 frames of each video were compressed, with I-frames inserted every 15 frames. CABAC and B-frames were disabled to lower the encoding complexity (these features are not likely to be used in digital camcorders). In the ME process, two reference frames were used and the search range was ±16 integer pixels.

A. Demosaicking Algorithm Choice

Here we evaluate the effectiveness of different demosaicking algorithms for use in our modified MC scheme.

Four different demosaicking algorithms were tested; bilinear interpolation, an edge-sensing method by Hamilton and Adams using Laplacian second order correction terms [4] (referred to as Laplacian in this paper), the "Primary-Consistent Soft-Decision" (PCSD) [7] method by Wu and Zhang, and a state-of-the-art method based on estimating luminance from the CFA data [9]. These algorithms vary in complexity and the quality of image they produce; bilinear interpolation has very low complexity but gives the poorest image quality. Laplacian demosaicking is intermediate in both image quality and complexity. PCSD gives higher image quality than Laplacian, but at much greater computational cost. The luminance based method gives the highest image quality (in terms of mean-square error), but also has much higher complexity than Laplacian demosaicking.

Rate distortion curves obtained using different demosaicking algorithms for MC are shown in Fig. 9. In Fig. 9, the peak signal-to-noise ratio (PSNR) of the CFA data is measured (as opposed to measuring quality after demosaicking has been performed). Results are shown for our proposed MC scheme with each demosaicking algorithm, as well as compressing the CFA video with a standard H.264 encoder in 4:2:2 sampling mode (our first proposed method). The results for both videos show there is substantial benefit in using our modified MC scheme. The bit rate reduction for the *CrowdRun* sequence at 31.4 dB composite PSNR (CPSNR) is about 4%, 10%, 10%, and 12% using bilinear, Laplacian, PCSD and luminance demosaicking, respectively. Greater bit rate reduction is obtained on the *OldTownCross* sequence, up to 15% using bilinear, 43% using Laplacian, 44% using PCSD, and 48% using luminance demosaicking.

These results show that the more advanced demosaicking schemes provide significant bit rate reductions over simple demosaicking (bilinear). This is expected, as simple bilinear demosaicking does not take advantage of inter-color correlation in the interpolation process. Laplacian demosaicking provides bit rate reductions almost as great as the more complex PCSD and luminance methods, making it well-suited for use in the MC process in applications where the complexity should be kept low.

B. Quality Comparison Against Demosaick-first Approach

In this section we compare our two methods against the conventional demosaick-first approach. When comparing to the demosaick-first approach, the quality of the final RGB video, after both compression and demosaicking have been performed, needs to be measured. The quality measure used here is the CPSNR, which has been used in previous work on compressing CFA data [13], [19]. The CPSNR is calculated as the standard PSNR, but with the mean-square error evaluated across the red, green and blue color channels. The CPSNR for one frame of video with 8 bit data is given by

$$\mathrm{CPSNR} = 10\log\left(\frac{255^2}{\frac{1}{3MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=1}^{3}\left(I_{\mathrm{comp}}(i,j,k)-I_{\mathrm{ref}}(i,j,k)\right)^2}\right) \tag{4}$$

where k denotes the color component (R, G, or B), $I_{\mathrm{comp}}(i,j,k)$ is the compressed video after demosaicking, and $I_{\mathrm{ref}}(i,j,k)$ is the reference video against which quality is measured. The reference is taken as the video obtained by demosaicking the CFA data without any compression. Thus, the reference video represents the best quality that can be obtained from the CFA data with a given demosaicking algorithm. The CPSNR for a video is taken to be the average CPSNR of the frames in the video.

The proposed methods were compared against the conventional demosaick-first approach. The demosaick-first results were obtained by demosaicking the CFA data, converting from RGB to YUV and compressing with standard H.264. Results are presented for the demosaick-first approach using both YUV 4:2:0 and YUV 4:2:2 format during compression. Plots of CPSNR versus bit rate for the two test videos are shown in Fig. 10, using each of the demosaicking methods described in the previous section of this paper. Regardless of the demosaicking method used, both proposed methods outperform the conventional demosaick-first approach at high bit rates. This is primarily because the proposed methods compress data one third the raw size of the demosaick first approach.

At low bit rates the demosaick-first approach gives better results than our proposed methods. There are several reasons for this. Demosaicked data has higher inter-pixel correlation than

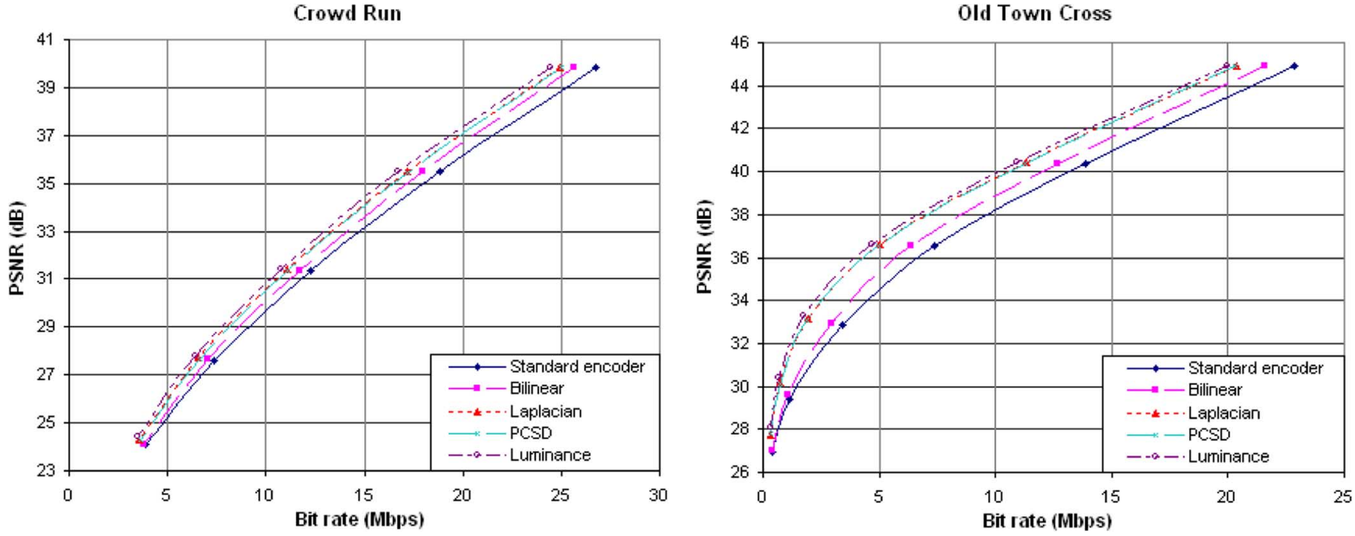


Fig. 9. Plots of PSNR (of the CFA data) versus bit rate.

CFA data, which makes transform coding more efficient. Correlation between the RGB colour channels is exploited through the use of the YUV transform when compressing full colour demosaicked data, but not when compressing CFA data. Also, highly compressing the CFA data removes detail necessary for demosaicking, and may introduce blocking artifacts which are interpreted as edges in the demosaicking process. Consequently demosaicking does not work well on highly compressed data. This problem obviously does not arise when demosaicking is performed prior to compression. In spite of these issues, at low compression ratios the smaller number of samples compressed results in the proposed compress-first approaches giving better performance.

The results obtained here (the compression first approach being more efficient at high bit rates) mirror the results obtained in an analysis of CFA still image compression in [26]. Analytical models for the quality of the demosaick-first and compress-first processing chains are presented in [26] for the case of JPEG and JPEG2000 based CFA image compression. The compression error model used in that paper assumes subband coding, which prevents the model there being directly applicable to H.264 video compression (which is not subband based). However, [26] still reveals some insight into the performance of the two processing chains when applied to video compression. A main conclusion in [26] is that the relative performance of the processing chains (demosaick-first and compression first) depends upon the coding gains of the transform when applied to the demosaicked data and the CFA data. The coding gain is higher for the demosaicked data, because the demosaicked frames have more inter-pixel and inter-colour correlation. At low bit rates the higher coding gain of the transform on demosaicked frames results in the traditional demosaick-first approach being superior, at high bit rates the smaller input data size of the compression-first approach results in it giving better performance. The situation is similar with H.264 video compression; intra-prediction, MC and integer transform perform better on the demosaicked data than the CFA data, resulting in the demosaick-first approach being better at high compression ratios. As is the case with still images, at low compression ratios the smaller number of samples compressed results in the compress-first approach being superior.

Comparing Method 2 (Modified MC) against Method 1 (standard H.264) gives the gain due to the proposed modified MC scheme. For both videos there is significant bit rate reduction obtained by using the proposed MC, particularly in the *OldTownCross* video which contains camera motion.

The graphs in Fig. 10 show that for the more advanced demosaicking methods (PCSD, Luminance), the bit rate at which the proposed methods outperform the demosaick-first approach is higher. This is because the advanced demosaicking methods are more sensitive to compression errors then the simple ones. The advanced methods are much better at preserving fine detail in images, if the detail has been corrupted in the compression process the advanced demosaicking provides less benefit. Consequently, the more advanced the demosaicking method is, the more advantage there is to applying it to uncompressed data. Using more advanced demosaicking also results in Proposed Method 2 (modified MC) providing more benefit over Method 1 (Fig. 10), because the more advanced demosaicking methods provide better prediction in the MC process. It should be noted that the main advantage of Method 1 is that it uses standard H.264, which makes it easier to incorporate into new applications.

C. Complexity

In the conventional demosaick-first approach, a video of size $M \times N$ is compressed, usually in YUV 4:2:0 format, resulting in 1.5MN total samples. In either of our proposed methods, a video of size $(M/2) \times N$ is compressed in 4:2:2 format, which requires MN samples. Hence, our proposed methods require two-thirds the number of samples to be compressed as the demosaick-first approach. Although the input video size is smaller in our proposed methods, the video will have more detail (high-frequency image content) as each frame has effectively been downsampled by a factor of two in the vertical direction.

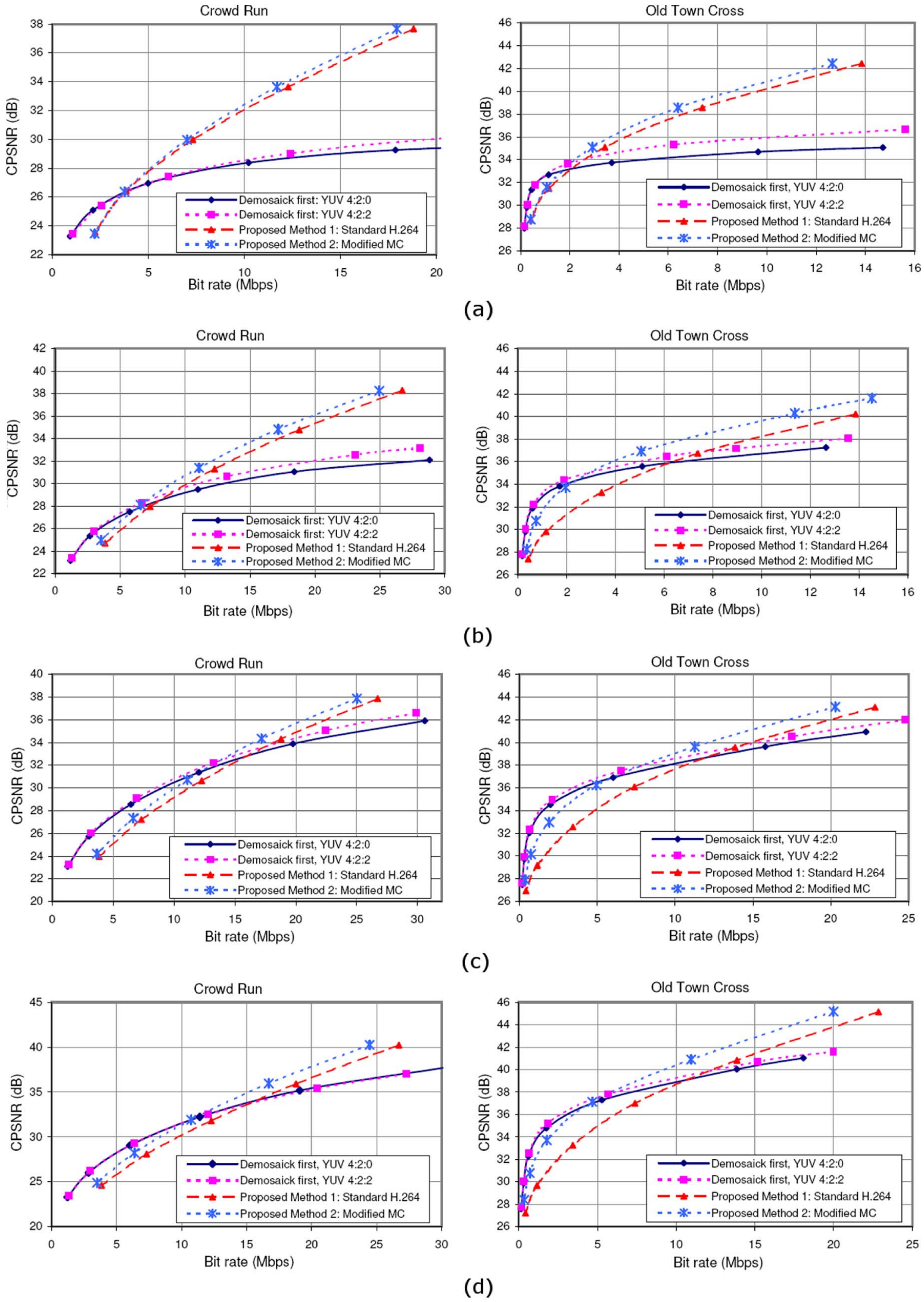


Fig. 10. Plots of CPSNR (measured after compression and demosaicking) versus bit rate obtained with different demosaicking methods. (a) Bilinear. (b) Laplacian [4]. (c) PCSD [7]. (d) Luminance estimation [9].

The most computationally expensive operation of an H.264 video encoder is ME, which can account for about 65% of the encoding time [27]. The complexity of ME varies greatly based on the algorithm used. If a full-search is used, the complexity of ME in either of our proposed methods will be about half that of the demosaick-first approach, since the luma channel has half the size. If a fast, content adaptive ME method is used, the ME complexity reduction obtained by using our methods will be variable. Other significant functions that contribute to encoder complexity are intra-prediction, interpolation, transform and quantization, loop filtering and entropy encoding [27]. With the exception of interpolation, the computational complexity of these functions varies depending on the video content. Generally video with more detail requires more computations. For these functions, our proposed methods will require fewer computations than the demosaick-first approach, but not by a factor of two-thirds since the smaller video will contain more detail.

The number of calculations for interpolation varies in our two methods. In Method 1 (standard H.264), luma interpolation requires half the number of computations of the demosaick-first approach, and chroma interpolation requires the same number of operations, due to the relative channel sizes. In Method 2 (Modified MC), the luma interpolation filter is applied to all of the red, green and blue channels. Therefore, the same number of luma

TABLE I
BIT RATE COMPARISON WITH METHOD IN [19]

Video	CPSNR (dB)	Bit-rate (Kbps)		
		Method in [19]	Proposed Method 1: Standard Encoder	Proposed Method 2: Modified MC
Foreman	29.4	1360	246	182
	32.5	2080	478	350
	36.0	2780	882	664
Carphone	27.2	812	100	90
	30.0	1154	194	166
	34.5	1850	458	370

interpolation calculations are required as in the demosaick-first approach, because the combined size of the R, G, and B channels is the same as the size of the luma channel in the demosaick-first approach. However, no chroma interpolation calculations are required in Method 2. Therefore, fewer interpolations operations are required in both proposed methods than in the demosaick-first approach.

In Method 1 (using standard H.264), demosaicking does not have to be performed at the encoder, which saves some computations relative to the demosaick-first approach. In Method 2, demosaicking is performed at the encoder, so the same number of demosaicking calculations are required as in the demosaick-first approach.

The most time consuming operations in H.264 video decoding are loop filtering, interpolation, inverse transform and quantization, and entropy decoding [28]. With the exception of interpolation, the amount of computations required for these operations is highly content and bit rate dependent. In general, videos with more detail require more computations. Hence, the amount of computations required for the loop filter, inverse quantization and entropy decoding will be lower in the proposed method due to the smaller frame size, but not by a factor of two-thirds because the video will have more detail. Our proposed methods will require two-thirds the number of inverse transform calculations, because the amount of calculations required to perform the inverse transform is not content dependent. The relative number of interpolation calculations is the same at the encoder and decoder; so our proposed methods require fewer interpolation calculations than the demosaick-first approach at the decoder as well. In either of our proposed methods, the decoder will have to perform demosaicking, which it would not in the demosaick-first approach. The decoder complexity of our proposed methods relative to the demosaick-first approach will depend on the video coding options used and the demosaicking algorithm.

On average, H.264 decoding of a QCIF sized frame (176×144) requires 30-40 million RISC operations [29], which corresponds to about 1200–1600 operations per pixel. By comparison, the Laplacian demosaicking method in [4] requires about 60 operations per pixel (the exact number will be dependent on the hardware and software implementation details). Hence, if a computationally efficient demosaicking method is used, demosaicking will only take a small fraction, around 4%–5%, of the decoding computations. This means that both proposed methods will have lower decoder complexity than the demosaick-first approach, due to the smaller frame size of the encoded video.

D. Comparison to Method in [19]

In order to compare our proposed methods against the method in [19], we use two standard test videos, foreman and carphone, for which results are presented in [19]. These videos are QCIF resolution (176×144) and sampled at 30 fps. Laplacian demosaicking is used in these tests to be consistent with [19]. Both videos were compressed at three quality levels and the resulting bit rates are summarized in Table I. The results for the method in [19] are taken directly from that paper. Both of our methods give much lower bit rates than the method in [19]. The bit rate reductions achieved relative to [19] range from 68% to 83% for our first method using standard H.264, and from 76% to 90% for our second method using modified MC. Because are methods are based on H.264, which is a highly optimized, efficient video coding standard, they achieve much better compression than the custom built method in [19].

V. CONCLUSION

In this paper, two methods are proposed for compressing Bayer pattern CFA video prior to demosaicking. Our first method involves separating the CFA data into arrays of green, blue and red samples and compressing in 4:2:2 sampling with standard H.264. In our second method, a modified MC scheme is also used, where demosaicking is performed on the references frames in the encoder and decoder in order to alleviate problems due to aliasing in the CFA data. Both proposed methods give better compression efficiency than the conventional demosaick-first approach at high bit rates, and much better performance than the only previously proposed method for compressing CFA video prior to demosaicking.

REFERENCES

[1] B. E. Bayer, "Color Imaging Array," U.S. Patent 3 971 ,065, Jul. 20, 1976.

[2] R. Lukac and K. N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Trans. Consum. Electron.*, vol. 51, no. 11, pp. 1260–1267, Nov. 2005.

[3] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schaffer, and R. M. Murserau, "Demosaicking: Color filter array interpolation," *IEEE Signal Process. Mag.*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[4] J. F. Hamilton and J. E. Adams, "Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera," U.S. Patent 5 629 734, May 13, 1997.

[5] B. Gunturk, Y. Altunbasak, and R. Mersereau, "Color plane interpolation using alternating projections," *IEEE Trans. Image Process.*, vol. 11, no. 9, pp. 997–1013, Sep. 2002.

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

[7] X. Wu and N. Zhang, "Primary-consistent soft-decision color demosaicking for digital cameras," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1263–1274, Sep. 2004.

[8] J. W. Glotzbach, R. W. Schafer, and K. Illgner, "A method of color filter array interpolation with alias cancellation properties," in *Proc. IEEE Int. Conf. Image Process.*, 2001, vol. 1, pp. 141–144.

[9] N. Lian, L. Chang, and Y. P. Tan, "Improved color filter array demosaicking by accurate luminance estimation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2005, vol. 1, pp. 41–44.

[10] X. Wu and L. Zhang, "Color video demosaicking via motion estimation and data fusing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 2, pp. 231–240, Feb. 2006.

[11] X. Wu and L. Zhang, "Improvement of color video demosaicking in temporal domain," *IEEE Trans. Image Process.*, vol. 15, pp. 3138–3151, Oct. 2006.

[12] S. Y. Lee and A. Ortega, "A novel approach of image compression in digital cameras with a Bayer color filter array," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2001, vol. 3, pp. 482–485.

[13] 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. 1448–56, Apr. 2003.

[14] T. Toi and M. Ohta, "A subband coding technique for image compression in single CCD cameras with bayer color filter arrays," *IEEE Trans. Consum. Electron.*, vol. 45, no. 1, pp. 176–80, Jan. 1999.

[15] A. Bruna, A. Buemi, F. Vella, and A. Vitali, "A low cost algorithm for CFA data compression," in *Proc. Int. Conf. Consum. Electron. Dig. Techn. Papers*, Jan. 2006, pp. 385–386.

[16] A. Bazhyna, A. Gotchev, and K. Egiazarian, "Lossless compression of Bayer pattern color filter arrays," in *Pro. SPIE and IS&T Electronic Imaging, Image Processing: Algorithms and Systems IV*, San Jose, CA, Jan. 2005, pp. 378–387.

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

[18] L. Zhang, X. Wu, and P. Bao, "Real time lossless compression of mosaic video sequences," *Real Time Imaging*, vol. 11, pp. 370–377, 2005.

[19] F. Gastaldi, C. C. Koh, M. Carli, A. Neri, and S. K. Mitra, "Compression of videos captured via bayer patterned color filter arrays," in *13th Eur. Signal Process. Conf.*, Antalya, Turkey, 2005.

[20] T. Wedi and H. G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 577–587, Jul. 2003.

[21] T. Wedi, "Adaptive interpolation filters and high-resolution displacements for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 484–491, Apr. 2006.

[22] D. Alleysson and S. Süsstrunk, "Aliasing in digital cameras," in *Proc. SPIE EI Newsletter, Special Issue on Smart Image Acquisition Process.*, 2004, vol. 14, pp. 1–8.

[23] J. Adams, K. Parsulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro*, vol. 18, no. 6, pp. 20–29, Nov./Dec. 1998.

[24] H. J. Trussell and R. E. Hartwig, "Mathematics for demosaicking," *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 485–492, Apr. 2002.

[25] L. Haglund, "The SVT high definition multi format test set," Sveriges Television,, Feb. 2006 [Online]. Available: ftp://vqeg.its.bldrdoc.gov/HDTV/SVT_MultiFormat/

[26] 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. 3261–3278, Nov. 2006.

[27] A. Hallapuro and M. Karczewicz, *Complexity Analysis of H.26L,* 2001, ITU-T SG16 Doc. VCEG-M50.

[28] V. Lappalainen, A. Hallapuro, and T. D. Hamalainen, "Complexity of optimized H.26L video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 717–725, Jul. 2003.

[29] C. Xu, T. M. Le, and T. T. Tay, "H.264/AVC codec: Instruction-level complexity analysis," in *Proc. IASTED Int. Conf. Internet Multimedia Systems, Applications (IMSA 2005)*, Honolulu, Hawaii, Aug. 2005, pp. 341–346.

Colin Doutre (S'06) was born in Montreal, QC, Canada, in 1982. He received the B.Sc. degree in electrical engineering from Queen's University, Kingston, ON, Canada, in 2005 and the M.A.Sc degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2007, where he is currently working toward the Ph.D. degree.

His research interests include video coding, multiview video, and digital cameras.

Mr. Doutre has received numerous awards and scholarships, including the Professional Engineers of Ontario Gold Medal and the Natural Sciences and Engineering Research Council of Canada (NSERC) Canada Graduate Scholarship.

Panos Nasiopoulos (M'96) received the B.S. degree in physics from Aristotle University, Thessaloniki, Greece, and the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada.

He is an Associate Professor with the Department of Electrical and Computer Engineering, University of British Columbia (UBC), Vancouver, BC, Canada, the holder of the MidNet Professorship in Digital Multimedia, and the current Director of the Master of Software Systems Program at UBC. Before joining UBC, he was the President of Daikin Comtec US. Daikin had worked together with Toshiba to introduce the first complete DVD solution to the world in 1996 and developed all the software components of the DVD system. He was voted as one of the most influential DVD executives in the world. He is recognized as a leading authority on DVD and multimedia and has published numerous papers on the subjects of digital video compression and communications. He has organized and chaired numerous conferences and seminars and he is a featured speaker at multimedia/DVD conferences worldwide.

Dr. Nasiopoulos has been an active member of the DVD Association and SPMTE as well as the ISO/ITU and In-Flight-Entertainment committees.

Konstantinos N. (Kostas) Plataniotis (S'90–M'82–SM'03) received the B.Eng. degree in computer engineering from University of Patras, Patras, Greece, in 1988 and the M.S. and Ph.D. degrees in electrical engineering from Florida Institute of Technology (Florida Tech), Melbourne, in 1992 and 1994, respectively.

He is an Associate Professor with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, an Adjunct Professor with the School of Computer Science at Ryerson University, Toronto, ON, Canada, and a member of The University of Toronto's Knowledge Media Design Institute. His research interests include biometrics, communications systems, multimedia systems, and signal and image processing. He is a registered Professional Engineer in the province of Ontario, and a member of the Technical Chamber of Greece.

Dr. Plataniotis is an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE SIGNAL PROCESSING LETTERS. He is a member of the IEEE TAB Committee on Biometrics, a Guest Editor for the forthcoming EURASIP *Journal on Advances in Signal Processing*, special issue on "Advanced Signal Processing and Pattern Recognition Methods for Biometrics" and co-editor of the IEEE Press volume entitled *Biometrics: Theory, Method and Applications* to be published in 2008. He is the 2005 recipient of IEEE Canada's Outstanding Engineering Educator Award "for contributions to engineering education and inspirational guidance of graduate students" and the co-recipient of the 2006 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award for the published in 2003 paper entitled "Face Recognition Using Kernel Direct Discriminant Analysis Algorithms".