RI-002

Post-processing of YCbCr4:2:0 Compressed Image for Color Enhancement

Yuji Itoh† and Tsukasa Ono†

1. INTRODUCTION

Most of the image and video compression coding standards such as JPEG [1] employ YCbCr color space, where the 'Y' component is referred to as luminance (or luma) while Cb and Cr are chrominance (or chroma) components. The highest resolution format of YCbCr color space is YCbCr4:4:4 (or just 4:4:4), which means that for every 4 samples of Y, there are also 4 samples of 'Cb' and 4 samples of 'Cr'. Meanwhile, the most commonly used format is YCbCr4:2:0 (or 4:2:0), where there are four *Y* samples for every chrominance (Cb and Cr) sample. Note that YCbCr format is often referred to as YUV, and that Cb and Cr are represented by *C* in case the color classification is not needed.

Fig. 1 schematizes the block diagram of a general image coding system, in which the image data format at each stage is noted. The compressed bit stream is deciphered by an image decoder into 4:2:0 format, which is up-converted into 4:4:4 format at the up-sampler. Then, the resultant 4:4:4 image is transformed to RGB (Red, Green, Blue) space for displaying and to CMY (Cyan, Magenta, Yellow) space for printing. We only deal with the RGB color space in this paper because CMY images can be directly and uniquely transformed from RGB images.

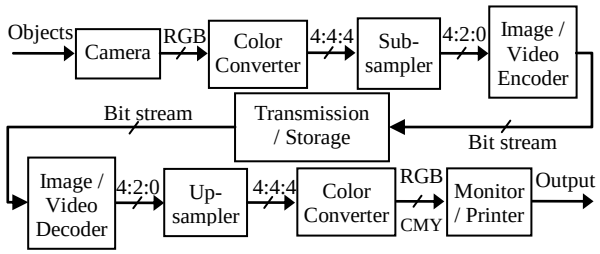

Fig. 1: Block diagram of typical image coding system.

It should be noted that the chrominance up-sampling is beyond the scope of the compression coding standards. Thus, a lot of technologies intended for the chrominance up-sampling have been proposed. In the conventional methods, an interpolation filter, which characterizes the up-sampling algorithm, is applied indiscriminately (i.e. non-adaptively) to entire image. The most widely used interpolation filters include nearest neighbor, bi-linear, *sinc* and *spline*. No matter how complicated the chosen filter is, the up-sampled image often suffers from aliasing, edge blurring and other artifacts. To avoid artifacts and preserve fine details, adaptive up-sampling techniques such as a multi-channel approach [2] have been introduced. In general, these adaptive techniques are based on segmenting an image dynamically into homogeneous region, and then apply some interpolation filter within the homogeneous regions. Many adaptive techniques [3]-[9] that address the 4:2:0 chrominance up-sampling have been proposed so far. They mostly exploit the inter-color (or inter-channel, inter-component) correlation in the YCbCr color space. We also proposed a unique approach [10] that exploits the correlation between RGB components (rather than the correlation between the luminance and chrominance channels).

† Tsukuba University of Technology

All these methods work well for non-compressed images. However, they do not take account of the fact that the inter-color correlation based processing is hampered by coding noises (or coding artifacts) when applying to compressed images. In this context, we propose a post-processing intended to generate high-fidelity RGB image from compressed image in the 4:2:0 format.

The rest of the paper is organized as follows. Section 2 explains the state-of-the-art methods in the chrominance up-sampling related fields. Section 3 describes the concept and embodiment of the proposed algorithm, which is evaluated in Section 4. Finally we will make some concluding remarks in Section 5.

2. REVIEW OF STATE-OF-THE-ART METHODS

This section investigates the state-of-the-art methods that address the 4:20 chrominance up-sampling. In the last decade, several approaches that exploit the correlation between the luminance and chrominance channels have been proposed. Jiang [5] and Bartkowiak [6] revealed that the correlation between gradient (i.e. gray scale difference in each color channel) images is much higher than the correlation between raw color components such as RGB. Thus, they proposed a gradient based scheme based on this observation. In the inter-channel correlation based approaches, the chrominance up-sampling is done by interpolating a missing pixel (i.e. pixel to be interpolated) using intra-channel input pixels (i.e. *C* channel) and inter-channel input pixels (i.e. *Y* channel). Suppose that there is the test sample pattern as shown in Fig. 2.

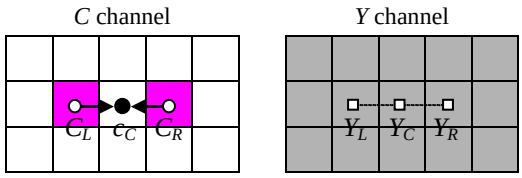

Fig. 2: A test sample pattern. This illustrates which pixels are involved in the interpolation: i) missing pixel (black circle), ii) intra-channel input pixels (white circle), and iii) inter-channel input pixels (white rectangular).

According to [5], [6], the missing pixel is obtained by:

$$c_C = \begin{cases} \dfrac{C_L + C_R}{2} & \text{if } Y_C < \min(Y_L, Y_R) \text{ or } \max(Y_L, Y_R) < Y_C \\ \dfrac{|Y_R - Y_C| \cdot C_L + |Y_L - Y_C| \cdot C_R}{|Y_L - Y_C| + |Y_R - Y_C|} & \text{otherwise} \end{cases} \quad (1)$$

This would work well if the cross-correlation between *Y* and *C* channels is high enough. In reality, however, the *Y*-*C* correlation coefficients (absolute values) are less than 0.3 whereas the *G*-*B* and *G*-*R* correlation coefficients usually reach 0.9, which are shown in the experiments later (Table 1).

In our recent work [10], we proposed a novel approach that exploits the correlation between the RGB components (rather than the luma-chroma correlation). It worked well for non-compressed images. Then, we need to take some counter

measure against the coding noises that attenuate the inter-color correlation based processing when applying to compressed images. To this end, we incorporate a *local map* based noise suppression technique [11] that the author originally developed, which was adopted as MPEG-4 de-ringing filter [12]. In the experiment later, we will also test MPEG-4 de-blocking filter [12] and TML post-processing filter [13] as noise suppression technique for benchmarking purposes.

3. PROPOSED POST-PROCESSING ALGORITHM

First, we explain the scheme to convert the color space between RGB and YCbCr, which is designated in the JPEG JFIF format [14].

$$\begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.500 \\ 0.500 & -0.4187 & -0.0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & \end{pmatrix} \begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} 1 & & \alpha \\ 1 & \chi & \delta \\ 1 & \beta & \end{pmatrix} \begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} \tag{3}$$

We noticed in [10] that the chrominance up-sampling problem resembles the CFA interpolation in the sense that both synthesize the missing color pixels in regularly sub-sampled image plane. In the CFA interpolation arena, many schemes reported in the literature employ so-called color difference model (CDM) [15] that exploits the inter-color correlation in the RGB domain.

Here we revisit the test case of the chrominance up-sampling depicted in Fig. 2. The question we are going to solve is to interpolate the sample a_C, which is illustrated in Fig. 4, using the neighboring a_C pixels and G (Green) pixels already available. Note that a (or A) represents R and B component depending on the occasion, and that the samples already available are in upper-case whereas those being interpolated, or to be interpolated are in lower-case.

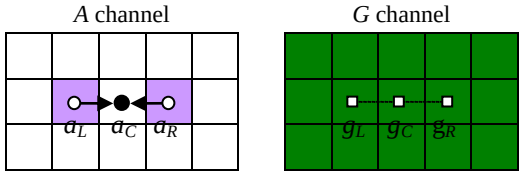

Fig. 3: The test sample pattern that corresponds to Fig. 2.

With the CDM-based CFA interpolation in [15], the missing color pixel a_c is calculated by:

$$a_c = \frac{a_L + a_R}{2} + \frac{2 \cdot g_C - g_L - g_R}{2} \tag{4}$$

The calculation of the missing chrominance pixel c_c is derived from Eqs. (3) and (4):

$$\begin{aligned} c_c &= (a_c - Y_C)/\gamma \\ &= \left(\frac{a_L + a_R}{2} + \frac{2 \cdot g_C - g_L - g_R}{2} - Y_C \right)/\gamma \\ &= \left[\frac{(a_L - Y_L) + (a_R - Y_R)}{2} \right]/\gamma + \left[\frac{2 \cdot (g_C - Y_C) - (g_L - Y_L) - (g_R - Y_R)}{2} \right]/\gamma \\ &= \frac{C_L + C_R}{2} + \left[\frac{2 \cdot (g_C - Y_C) - (g_L - Y_L) - (g_R - Y_R)}{2} \right]/\gamma \end{aligned} \tag{5}$$

$$\gamma = \begin{cases} \alpha & \text{if } a \in R \text{ components} \\ \beta & \text{if } a \in B \text{ components} \end{cases}$$

We examined the second term of the last expression in Eq. (5) using the 4:4:4 images and the converted RGB images. It was, then, found that the second term of Eq. (5), where g samples are not available yet though, usually becomes very small for the natural scene images. This suggests that the first term of Eq. (5) be a pertinent estimator. Thus, the missing chrominance value denoted by c_c' is given by:

$$c_c' = \frac{C_L + C_R}{2} \tag{6}$$

This is a plain intra-channel averaging of the neighboring pixels, rather than the complicated adaptive interpolation employed in the state-of-the-art methods. Now the G pixel values can be calculated according to Eq. (3) using the luminance samples and the interpolated chrominance samples calculated by Eq. (6).

In the next step, we obtain R and B pixel values by applying the CDM based interpolation technique. To this end, R and B pixel values at regularly sampled position that is co-sited with G pixel need to be obtained beforehand. We can calculate such R and B pixel values by Eq. (3) directly if the 4:2:0 sampling position is "co-sited". For other 4:2:0 sampling positions such as "centered", the co-sited chrominance values must be calculated by some method, e.g., bi-linear interpolation, using the 4:2:0 sampled chrominance signals. Then, the missing R and B pixel value is given by:

$$a_c = G_{C+} \sum_{(i,j) \in \zeta} w_{(i,j)} \cdot \left(A_{(i,j)} - G_{(i,j)} \right) \tag{7}$$

where ζ represents the neighboring pixel positions around the missing pixel, and $w(i,j)$ is the weighting coefficient of the interpolation filtering. Note that Eq. (7) is a general expression of the CDM-based interpolation.

We have recognized that finding an optimal filtering direction is the most critical task for the chrominance up-sampling. The proposed algorithm achieves this task using the local map. The map indices are obtained on a local map window (N by N block, N=8 for the experiments later) basis on Y plane with a threshold specific to the input Y sample data. In each local map window, a threshold value shall be determined first. Let *max* and *min* be the maximum and minimum gray scale values in a luminance block, respectively. We define the threshold denoted by τ, which is the middle of dynamic range:

$$\tau = (max + min)/2 \tag{8}$$

Now let $\lambda(i, j)$ denote map index at coordinates (i, j). The map index $\lambda(i, j)$ is determined based on whether the pixel value $Y(i,j)$ is greater than the threshold or not, which is given below.

$$\lambda_{(i,j)} = \begin{cases} 1 & \text{if } Y_{(i,j)} \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The map indices provide a very rough idea of the similarity (i.e., the pixels with the same indices generally have similar values). Once the map indices are obtained, an interpolation filter is applied to all the relevant pixels in the local map window (i.e., N by N block). In this way, we can significantly reduce the computational complexity, which is quantitatively expressed later in the experiments (Table 3).

Fig. 5 shows the block diagram of the de-ringing and interpolation schemes that are employed in the proposed algorithm. The schemes are basically three-fold: i) local map acquisition, ii) map-based de-ringing, and iii) map-based interpolation. It should be noted that the de-ringing and the interpolation are of the same structure, that is, both use the local map.

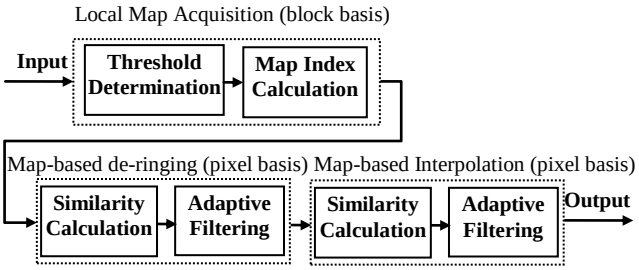

Fig. 4: Block diagram of interpolation algorithm.

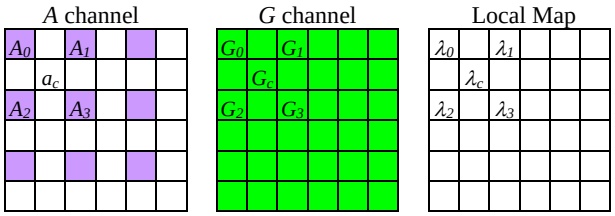

Fig. 5: An example case of missing *A* pixel interpolation. The arrangement of input *A*, *G* channel data and local map are illustrated.

An example case of a missing *A* pixel interpolation is shown in Fig. 6, where a_c is a missing pixel of the *A* channel collocated with G_c. In the following equations, θ represents the similarity measure, and ζ denotes the local vicinity, which is equivalent to the input pixels, around the missing pixel (Fig. 8 depicts ζ in respective case). Note that the position index *0-3* in Fig. 6 corresponds to (*i*, *j*) within ζ. Now we calculate the missing *A* (*R* and *B)* pixel according to Eq. (10), in which respective directional component and omni-directional component denoted by ρ are added when θ=1 and θ=0, respectively.

$$a_C = G_C + \left\{ 4\rho + \sum_{(i,j)\in\varsigma} \left[\theta_{(i,j)} \cdot \left(A_{(i,j)} - G_{(i,j)} \right) + \bar{\theta}_{(i,j)} \cdot \rho \right] \right\} / 8$$
$$\theta_{(i,j)} = \begin{cases} 1 & if\ \lambda_{(i,j)} = \lambda_C \\ 0 & otherwise \end{cases} \qquad (10)$$
$$\rho = \frac{1}{4} \sum_{(i,j)\in\varsigma} \left(A_{(i,j)} - G_{(i,j)} \right)$$

Now we move onto the second topic, i.e. the de-ringing. The tasks are two-fold: i) calculation of similarity, and ii) adaptive filtering. As we discussed earlier, the de-ringing and the chrominance up-sampling use the same local map obtained according to Eqs. (8) and (9). Fig. 7 illustrates an example case of the de-ringing, where Y_c' is the output of the de-ringing and *max_diff* represents the parameter for the quality control. With the same notation as in Eq. (10) above, the de-ringing is performed according to Eq. (11). Note that the de-ringing precedes the chrominance up-sampling.

$$\begin{aligned} &if \quad Y > Yc + max\_diff \\ &\quad Yc' = Yc + max\_diff \\ &else\ if \quad Y < Yc - max\_diff \\ &\quad Yc' = Yc - max\_diff \\ &else \\ &\quad Yc' = Y \end{aligned} \qquad (11)$$
$$Y = \left\{ 4 \cdot Yc + \sum_{(i,j)\in\varsigma} \left[\theta_{(i,j)} \cdot Y_{(i,j)} + \bar{\theta}_{(i,j)} \cdot Yc \right] \right\} / 8$$
$$\theta_{(i,j)} = \begin{cases} 1 & if\ \lambda_{(i,j)} = \lambda_C \\ 0 & otherwise \end{cases}$$

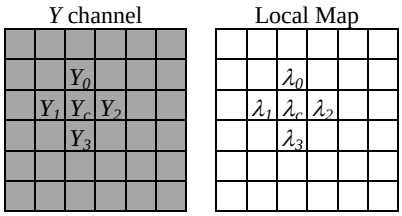

Fig. 6: An example case of de-ringing. The arrangement of input luminance data and local map are illustrated.

Fig. 8 shows the processing flow of the proposed algorithm after the local map is calculated according to Eqs. (8) and (9). The entire algorithm is done in multiple stages and on a block (i.e. the local map window) basis as below.

3.1 De-ring Y (A-stage)

Let ζ = {(*s*-1, *t*), (*s*, *t*-1), (*s*, *t*+1), (*s*+1, *t*)} in Fig. 8 (a), Y_c = *Y(s, t)* and λ_c = λ*(s, t)*. The luminance pixel at coordinates (*s*, *t*) is processed according to Eq. (11). This operation is applied to all the luminance pixels.

3.2 Intra C-1 (B-stage)

Let ζ = {(*s*-1, *t*-1), (*s*-1, *t*+1), (*s*+1, *t*-1), (*s*+1, *t*+1)} in Fig. 8 (b). The missing chrominance pixel at coordinates (*s*, *t*), denoted by *c(s,t)*, is interpolated according to Eq. (12).

$$c_{(s,t)} = \frac{1}{4} \sum_{(i,j)\in\varsigma} C_{(i,j)} \qquad (12)$$

3.3 Intra C-2 (C-stage)

Let ζ = {(*s*-1, *t*), (*s*, *t*-1), (*s*, *t*+1), (*s*+1, *t*)} in Fig. 8 (c). The missing chrominance pixel at coordinates (*s*, *t*), denoted by *c(s,t)*, is interpolated according to Eq. (12). The entire chrominance planes are filled up after this stage is applied to all the missing chrominance pixels. Thereafter, the full population of *G* plane and quarter population (i.e. only at sub-sampled positions) of *R* and *B* planes are calculated according to Eq. (3).

3.4 Inter RB-1 (D-stage)

Let ζ = {(*s*-1, *t*-1), (*s*-1, *t*+1), (*s*+1, *t*-1), (*s*+1, *t*+1)} in Fig. 8 (d), c_c = *c(s, t)* and G_c = *G(s, t)*. The missing *A* pixel at coordinates (*s*, *t*) is interpolated according to Eq. (10).

3.5 Inter RB-2 (E-stage)

Let ζ = {(*s*-1, *t*), (*s*, *t*-1), (*s*, *t*+1), (*s*+1, *t*)} in Fig. 8 (e). The missing *A* pixel at coordinates (*s*, *t*) is interpolated according to Eq. (10).

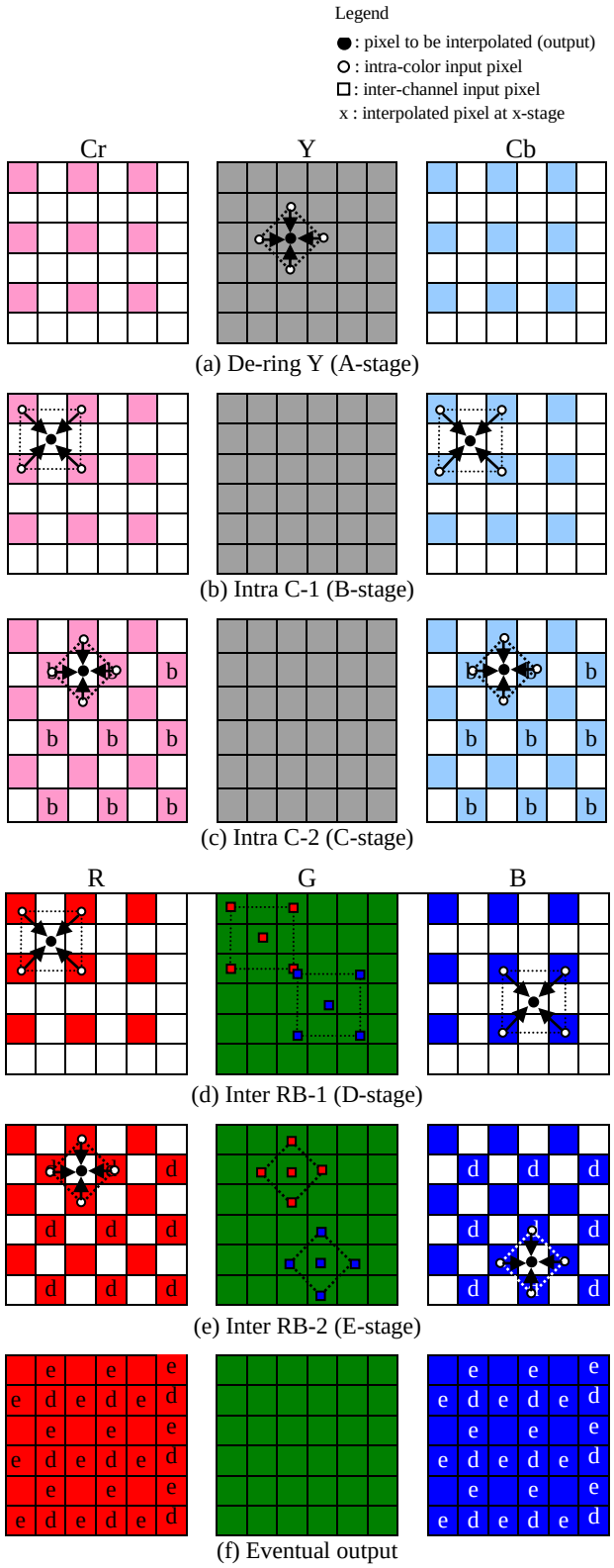

Fig. 7: Processing flow of proposed algorithm. The characters in a pixel represent the stage in which the pixel is interpolated, e.g., 'b' means that the pixel was interpolated in the B-stage.

4. EXPERIMENTS

We assess the performance of the method in [5], [6] and the proposed scheme with three kinds of the noise suppression techniques, and compare them quantitatively. First, the original image in the RGB color space is transformed into the YCbCr color space according to Eq. (2). Next, the 4:4:4 format image is decimated to the 4:2:0 format. We use "co-sited" sampling position for ease of comparison in the experiments since the additional filtering required for the other positions tend to smear the signals with extra artifacts (hence hinders the comparison of the processed images between the algorithms under test). The 4:2:0 format images are coded using JPEG [1] with adequate quantization tables, so that the *PSNR* (according to Eq. (14)) of *R* and *B* components falls between 30dB and 35dB for the mid-low quality range and the *PSNR* exceeds 35dB for the high quality range. Then, the decoded 4:2:0 image is processed with the noise suppression technique specified (if any). The quality control parameter of the de-ringing, i.e., *max_diff*, for JPEG is given by Eq. (13).

$$max\_diff\ (=QP) = Q_{avg}/8$$
$$Q_{avg} = \frac{1}{63}\sum_{i=1}^{63} Q_i \qquad (13)$$

where Q_i reperesents *i*-th entry (in zig-zag scanning order) of the quantization table of Y channel (i.e., Q_0 corresponds to DC component). The quantization parameter *QP* for the MPEG-4 de-blocking filter [12] and the TML post-processing [13] is derived in a similar manner, that is, empirically choosing the *QP*, which corresponds to *max_diff* of the de-ringing, that provides the best PSNRs with varying the denominator of (13)). Then, the chrominance signals in the processed 4:2:0 image are up-sampled to the 4:4:4 format by the method in [5], [6]. Thereafter, the 4:4:4 format image is re-transformed to the RGB color space according to Eq. (3). Meanwhile, the proposed method de-rings the luminance signals, and then converts the 4:2:0 format image to the full resolution RGB image. Finally, the *peak-signal to noise ratio* (PSNR) of each color component between the processed RGB image and the original RGB image is calculated by Eq. (14).

$$PSNR_k = 10 \cdot \log_{10} \frac{255^2}{\frac{1}{ST}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\left(o_{(s,t)k} - x_{(s,t)k}\right)^2} \qquad (14)$$

where $\boldsymbol{o}_{(s,t)} = [o_{(s,t)\mathrm{R}}, o_{(s,t)\mathrm{G}}, o_{(s,t)\mathrm{B}}]$ and $\boldsymbol{x}_{(s,t)} = [x_{(s,t)\mathrm{R}}, x_{(s,t)\mathrm{G}}, x_{(s,t)\mathrm{B}}]$ denote the RGB vectors at coordinates (*s*,*t*) of the original image and processed image (both images are of extension *S* by *T*), respectively. We use the Kodak test images shown in Fig. 9. These images are all in the RGB color space with 768 by 512 pixels, 8 bits/pixel/color.

Fig. 8: Test images used for experiments. These are Kodak test images #1 through #24, from left to right, from top to bottom.

Table 1: Simulation results (without noise suppression techniques).

Image Number	Cross-correlation Coefficients				High quality range, PSNR						Mid-low quality range, PSNR					
					Method in [5], [6]			Proposed			Method in [5], [6]			Proposed		
	Y-Cb	Y-Cr	G-B	G-R	G	B	R	G	B	R	G	B	R	G	B	R
1	-0.068	-0.527	0.991	0.861	40.46	34.20	35.14	40.69	36.16	36.23	35.44	32.58	32.89	35.64	33.51	33.53
2	-0.156	0.025	0.974	0.509	42.90	34.52	35.44	43.31	36.81	35.78	34.10	31.73	31.00	34.39	32.38	31.22
3	-0.386	-0.096	0.550	0.715	44.25	34.80	36.29	44.62	36.47	37.28	35.62	32.25	33.13	35.90	32.86	33.49
4	0.497	-0.471	0.957	0.596	42.30	34.59	35.26	42.78	36.22	35.72	34.05	32.11	30.99	34.31	32.57	31.27
5	-0.380	-0.125	0.903	0.898	37.26	32.52	33.78	37.43	33.92	34.36	35.04	31.29	32.31	35.29	32.36	32.88
6	-0.844	0.442	0.991	0.977	42.28	34.32	35.74	42.42	36.16	37.20	34.19	31.31	32.19	34.37	31.98	32.66
7	-0.067	0.043	0.907	0.830	43.16	34.50	35.90	43.52	36.61	36.98	35.07	31.54	32.54	35.39	32.22	33.08
8	-0.474	-0.061	0.974	0.966	37.94	32.96	34.11	38.07	34.10	34.81	34.69	31.33	32.16	34.87	32.07	32.66
9	-0.492	0.120	0.853	0.948	42.91	34.51	35.52	43.11	36.49	37.33	35.15	31.95	32.61	35.36	32.50	33.03
10	-0.592	-0.004	0.969	0.955	43.14	34.54	35.69	43.36	36.22	37.63	34.99	31.81	32.65	35.30	32.42	33.28
11	-0.002	-0.317	0.974	0.821	41.62	34.36	35.50	41.84	36.15	36.54	34.56	31.91	32.12	34.83	32.68	32.66
12	-0.089	-0.515	0.967	0.912	43.62	34.80	35.92	43.88	36.72	37.54	35.26	32.14	32.85	35.53	32.61	33.33
13	0.030	-0.185	0.964	0.982	37.10	32.45	33.88	37.17	33.74	34.66	33.59	30.46	31.79	33.75	31.24	32.28
14	-0.542	0.112	0.709	0.860	40.85	33.57	34.83	41.30	35.79	35.88	33.55	30.19	31.06	33.89	31.12	31.62
15	-0.076	-0.519	0.988	0.860	42.14	34.60	35.23	42.51	35.99	35.63	34.12	31.61	31.25	34.35	32.16	31.52
16	-0.389	0.042	0.950	0.984	44.91	35.18	36.36	45.07	37.14	38.10	36.21	32.80	33.73	36.41	33.50	34.21
17	-0.167	-0.308	0.978	0.985	42.39	34.20	35.79	42.47	35.98	36.57	36.35	32.40	33.76	36.61	33.09	34.23
18	-0.607	0.483	0.835	0.916	39.76	33.41	34.83	39.93	35.04	35.75	35.84	31.74	32.87	36.01	32.72	33.41
19	-0.279	0.299	0.915	0.967	41.36	34.25	35.48	41.45	35.98	36.24	34.92	31.79	32.66	35.12	32.53	33.00
20	0.104	-0.107	0.976	0.995	42.01	34.34	36.42	42.11	36.11	37.33	34.49	31.29	32.99	34.82	31.83	33.34
21	0.004	-0.237	0.915	0.895	41.86	34.14	35.53	41.97	35.46	36.70	34.47	31.23	32.46	34.70	31.88	32.90
22	0.063	0.058	0.874	0.870	41.65	34.00	35.10	41.79	35.66	35.86	34.59	30.96	32.05	34.73	31.73	32.45
23	-0.233	0.009	0.638	0.612	43.29	34.72	35.86	43.59	36.34	36.75	34.26	30.75	31.61	34.56	31.29	32.05
24	-0.047	-0.111	0.971	0.970	38.49	31.82	33.92	38.47	32.46	34.34	35.63	30.63	32.56	35.74	31.10	32.88
Average	0.2745	0.2173	0.9051	0.8702	41.57	34.05	35.31	41.79	35.74	36.30	34.84	31.58	32.34	35.08	32.26	32.79
(Difference between the two methods)								+0.22	+1.68	+0.99				+0.24	+0.69	+0.45

Table 2: Simulation results, PSNR of mid-low quality range (with noise suppression techniques).

Image Number	Method in [5], [6]			Method in [5], [6] + MPEG-4 de-blocking			Method in [5], [6] + TML post-processing			Proposed w/o de-ringing + MPEG-4 de-blocking			Proposed w/o de-ringing + TML post-processing		
	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R
1	35.44	32.58	32.89	35.44	32.59	32.89	35.44	32.58	32.89	35.49	33.35	33.37	35.49	33.35	33.37
2	34.10	31.73	31.00	34.10	31.75	30.95	34.26	31.82	30.92	34.13	32.22	31.08	34.30	32.30	31.02
3	35.62	32.25	33.13	35.65	32.32	33.20	35.43	32.33	33.08	35.66	32.74	33.33	35.43	32.76	33.22
4	34.05	32.11	30.99	34.14	32.17	31.00	33.99	32.01	30.87	34.16	32.46	31.16	33.99	32.27	31.04
5	35.04	31.29	32.31	35.04	31.29	32.31	35.04	31.29	32.31	35.12	32.20	32.72	35.12	32.20	32.72
6	34.19	31.31	32.19	33.88	31.21	32.01	33.63	31.35	31.77	33.89	31.68	32.32	33.63	31.85	32.06
7	35.07	31.54	32.54	35.20	31.60	32.64	34.78	31.48	32.56	35.23	32.12	32.90	34.80	31.98	32.83
8	34.69	31.33	32.16	34.69	31.33	32.16	34.69	31.33	32.16	34.74	31.95	32.55	34.74	31.95	32.55
9	35.15	31.95	32.61	35.18	32.05	32.70	34.95	32.11	32.66	35.18	32.37	32.91	34.95	32.44	32.87
10	34.99	31.81	32.65	34.97	31.90	32.75	34.70	31.94	32.72	34.97	32.24	33.08	34.69	32.29	33.03
11	34.56	31.91	32.12	34.24	31.77	31.95	34.16	31.92	31.92	34.26	32.31	32.30	34.17	32.47	32.25
12	35.26	32.14	32.85	35.25	32.19	32.89	35.10	32.17	32.94	35.25	32.45	33.12	35.09	32.42	33.16
13	33.59	30.46	31.79	33.59	30.46	31.79	33.59	30.46	31.79	33.61	31.13	32.15	33.61	31.13	32.15
14	33.55	30.19	31.06	33.39	30.10	30.98	33.05	29.99	30.85	33.47	30.86	31.36	33.11	30.77	31.17
15	34.12	31.61	31.25	34.17	31.70	31.24	34.09	31.74	31.14	34.19	32.05	31.40	34.09	32.09	31.29
16	36.21	32.80	33.73	35.78	32.67	33.52	35.51	32.74	33.32	35.78	33.16	33.82	35.52	33.23	33.60
17	36.35	32.40	33.76	36.07	32.32	33.67	35.63	32.20	33.63	36.08	32.84	33.87	35.62	32.70	33.86
18	35.84	31.74	32.87	35.84	31.74	32.87	35.84	31.74	32.87	35.88	32.60	33.29	35.88	32.60	33.29
19	34.92	31.79	32.66	34.71	31.76	32.52	34.54	31.84	32.33	34.72	32.30	32.70	34.54	32.41	32.50
20	34.49	31.29	32.99	34.61	31.35	33.10	34.54	31.48	33.03	34.61	31.70	33.18	34.54	31.82	33.11
21	34.47	31.23	32.46	34.22	31.14	32.33	34.06	31.29	32.27	34.22	31.60	32.57	34.06	31.76	32.50
22	34.59	30.96	32.05	34.35	30.88	31.97	33.96	30.88	31.87	34.36	31.51	32.18	33.97	31.49	32.11
23	34.26	30.75	31.61	34.61	30.97	31.88	34.49	30.90	31.80	34.65	31.36	32.09	34.52	31.26	32.01
24	35.63	30.63	32.56	35.63	30.63	32.56	35.63	30.63	32.56	35.63	31.06	32.72	35.63	31.06	32.72
Average	34.84	31.58	32.34	34.78	31.58	32.33	34.63	31.59	32.26	34.80	32.09	32.59	34.65	32.11	32.52
Improvement over the method in [5], [6]				-0.06	+0.00	-0.01	-0.21	+0.02	-0.08	-0.04	+0.52	+0.25	-0.20	+0.53	+0.18

Table 1 summarizes the results of the simulation without the noise suppression, where the PSNRs of both high quality and mid-low quality ranges are presented. The *Y-C* correlation coefficients and the *G-R* and *G-B* correlation coefficients of the original signals are also shown in the table. We observe in the results that the proposed method consistently outperforms the method in [5], [6] in every case. The improvements of *G* component are relatively marginal because the up-sampled Cb

and Cr components have less contribution to the derivation of the *G* component as shown in Eq. (3). As for the *R* and *B* components, the proposed method gains approximately 1.33 dB and 0.57 dB over the method in [5], [6] on average, in the high quality range and the mid-low quality range, respectively. We assume that the performance difference between the two methods comes from the difference of the inter-color correlation between the YCbCr domain and RGB domain as shown in Table 1. In other words, inter-color based processing will better work where the color components are highly correlated.

Table 2 focuses on the collaboration with the noise suppression techniques in the mid-low quality range. Note that the noise suppression techniques under test do not work at all for the high quality range due to the quality control parameters provided by Eq. (13). The MPEG-4 de-blocking and the TML post-processing do not help improve the quality (but rather decrease PSNR) for the method in [5], [6]. They work well for the proposed up-sampling method with regard to the *R* and *B* components, whereas they worsen the *G* component quality. The primary objective of the color enhancement consists in the quality improvement of the *R* and *B* components. The MPEG-4 de-blocking and the TML post-processing techniques accomplish it. However, those that deteriorate the *G* (or *Y*) component quality would not be accepted as post-processing for image and video in general.

We suppose that the de-blocking filters will be more useful for video, or at lower bit rates, where the blocking noises will be more dominant than the ringing noises. In addition, the noise suppression performance may be able to be enhanced by using the coding parameters such as macroblock type, while we apply the noise suppression techniques without utilizing them in this study. Overall, the proposed method provided the best performance among all the combinations of the chrominance up-sampling and the noise suppression techniques under test.

Table 3: Normalized computations per 2-by-2 pixels.

Operations / Algorithm	shift	add / subtract	compare	absolute	multiply	divide
Method in [5], [6]	6	84	42	24	40	12
+ MPEG-4 de-blocking*	22	141	63	35	40	12
+ TML post-processing	46	132	90	24	40	12
Proposed w/o de-ringing	18	100	40	0	10	0
+ MPEG-4 de-blocking*	34	157	61	11	10	0
+ TML post-processing	58	148	88	0	10	0
Proposed	26	124	64	0	10	0
Note *: Average of the two modes for the MPEG-4 de-blocking is added.						

Table 3 shows the normalized computations required for processing 2-by-2 pixels including the color space conversion with regard to primitive arithmetic operations. The proposed algorithm requires slightly larger number operations than the method in [5], [6]. As for the resource hungry operations, our method is free from division and needs smaller number of multiplications than the method in [5], [6]. These indicate that the proposed algorithm is as complicated as the method in [5], [6]. It should be noted that six out of eight chrominance pixels are not transformed to *R* or *B* pixels in the proposed method because only two pixels (one for *R*, one for *B*) are transformed, and the remaining six pixels (three for *R*, three for *B*) are interpolated in the RGB color space. It is noteworthy that the proposed algorithm, which yielded the best PSNRs, has an implementation advantage, that is, the local map can be shared with the proposed chrominance up-sampling process. This actually saved 18 compare operations per 2-by-2 pixels. In case the noise suppression filter is incorporated into the method in [5], [6], which does not help improve the PSNR though, the total resource requirements exceed that of the proposed method.

5. CONCLUSION

This paper has described a post-processing of the compressed images in the YCbCr4:2:0 format for color enhancement. The simulation results showed that the proposed algorithm outperforms the state-of-the-art technologies by 1.33 dB in the high quality range and 0.57 dB in the mid-low quality range with regard to the *PSNR* of red and blue components. It was also revealed that the proposed algorithm provided the best performance among all the combinations of the chrominance up-sampling and the noise suppression techniques under test. In addition, we confirmed that the proposed algorithm is comparable to, or even less complicated than the state-of-the-art technologies with regard to the resource requirements. This algorithm comprises the chrominance up-sampling and noise suppression technique, both of which employ the same local map. Thus, the proposed algorithm is able to work as an integrated post-processing. We will apply the same idea to the video coding (or at lower bit rates) and the up-sampling of YCbCr4:2:2 format image in the future.

REFERENCES

[1] ISO/IEC 10918-1:1994, "Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines," 1994.

[2] K. Tang, J. Astola, and Y. Neuvo, "Multichannel edge enhancement in color image processing," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 4, no. 5, Oct. 1994.

[3] Y. Itoh and N.-M. Cheung, "Inter-chrominance up-sampling algorithm for color image processing," *Proc. Int. Conf. on Signal Processing Applications and Technology*, #363, Oct. 2000.

[4] Y. Itoh, "Inter-chrominance up-sampling algorithm for color post-processing", *IEICE Trans.*, Vol. E86-D, No. 1, pp.146-149, Jan. 2003.

[5] H. Jiang, "Edge-adaptive chroma up-conversion," U.S. Patent 6 297 801, Oct. 2001.

[6] M. Bartkowiak, "Improved interpolation of 4:2:0 colour images to 4:4:4 format exploiting inter-component correlation," *Proc. 12th EUSIPCO*, pp. 581-584, Sep. 2004.

[7] A. Dumitras and F. Kossentini, "FANN-based video chrominance subsampling," *Proc. ICASSP'98*, pp. 1077-1080, 1998.

[8] G. Qiu and G. Schaefer, "High quality enhancement of low resolution colour images," *Proc. IEEE Int. Conf. on Image Proc. and Its Applications*, vol. 1, pp. 358-362, 1999.

[9] M. Zhao, P.M. Hofman and G. de Haan, "Content-adaptive up-scaling of chrominance using classification of luminance and chrominance data," *Proc. VCIP*, SPIE, pp. 721-730, Jan. 2004

[10] Y. Itoh and T. Ono, Up-sampling of YCbCr4:2:0 Image exploiting Inter-color Correlation in RGB Domain, *IEEE Trans. Consumer Electron*, vol. 55, no. 4, pp. 2204-2210, 2009.

[11] Y. Itoh, "Detail preserving noise filtering for compressed video," *IEICE Trans. Commun.*, vol. E79-B, no. 10, pp.1459-1466, Oct. 1996.

[12] ISO/IEC 14496-2:1999, "Information technology -- Coding of audio-visual objects -- Part 2: Visual," 1999.

[13] G. Bjontegaard, "H.26L Test Model Long Term Number 2 (TML-2) draft 0," ITU-T Q.15/SG16, Q15-I-57d0, Nov. 1999.

[14] E. Hamilton, "JPEG File Interchange Format Version 1.02," 1992.

[15] J. E. Adams Jr., "Design of practical color filter array interpolation algorithms for digital cameras," *Proc. SPIE*, vol. 3028, pp. 117-125, Feb.1997.