Improving Reversible Color-to-grayscale Conversion with Halftoning

Zi-Xin Xu and Yuk-Hee Chan*

Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong

Abstract: *Reversible color-to-grayscale conversion (RCGC) aims at embedding the chromatic information of a full color image into its grayscale version such that the original color image can be reconstructed in the future when necessary. Conventional RCGC algorithms tend to put their emphasis on the quality of the reconstructed color image, which makes the color-embedded grayscale image visually undesirable and suspicious. This paper presents a novel RCGC framework that emphasizes the quality of both the color-embedded grayscale image and the reconstructed color image simultaneously. Its superiority against other RCGC algorithms is mainly achieved by developing a color palette that fits into the application and exploiting error diffusion to shape the quantization noise to high frequency band. The improved quality of the color-embedded grayscale image makes the image appears as a normal image. It does not catch the attention of unauthorized people and hence the embedded chromatic information can be protected more securely.*

Keywords: *Reversible Color mapping, Information Hiding, Halftoning, Color Quantization, Color Palette, Noise Shaping*

1. Introduction

Color images are heavily and widely used nowadays in multimedia applications. In some applications, we allow the grayscale content of an image to open to the public but only disclose its color information to a group of privileged people (e.g. the private digital painting database of the Louvre Museum of Paris, France) [1-4]. Sometimes, due to some practical constraints, we are forced to deliver or present an image in grayscale temporarily but we need its color information in the future[5-8]. In either scenario, it would be desirable to convert a full color image into a grayscale image with the chromatic information embedded inside such that we can reverse the color-to-grayscale conversion afterwards to reconstruct a color image which is close to the original. A reversible color-to-grayscale conversion supports the reversion purposely.

When an image is delivered electronically, it is possible to extract the chrominance information from the image, encrypt it and then keep it as side information for future use. However, this approach may make the image not compatible with a conventional format. Since the chrominance information is not self-contained in the image, none of it can be recovered without the side information. Worst still, it makes the image file suspicious as non-privileged people can easily find some mysterious data not interpretable, which is not acceptable when we do not want to trigger any unbearable or unpredictable consequences. In contrast, if the chrominance information is directly embedded in the grayscale image itself, the aforementioned problems can be automatically solved.

The basic idea of reversible color-to-grayscale conversion (RCGC) is to embed the chromatic information of a color image in its grayscale version. Since it involves information embedding, a naive approach is to compress the chromatic information and then embed it as binary data into the grayscale version with a reversible data hiding algorithm such as [9-11]. However, this approach does not provide a good performance in practical situations because the color information of a natural image increases with the image size and it is too rich for a reversible data hiding algorithm to handle. For example, even after a 4:2:0 chrominance subsampling process [12] is performed, the average entropy of the chrominance content of the color images in the Kodak set [13] is 2.9 bits per pixel (bpp), which still significantly exceeds the capacity of state-of-art reversible data hiding algorithms, and hence a further compression is required. The chromatic quality of the reconstructed color image can be degraded remarkably at a high compression ratio. Dedicated RCGC algorithms are hence needed to address this specific issue.

State-of-art RCGC algorithms are either based on vector quantization (VQ) technique [1-4,14] or subband embedding (SE) technique [5-8,15-17]. A VQ-based algorithm generates a palette of 256 colors with a clustering algorithm such as [18,19], quantizes each pixel color of the color image to one of the palette colors, and then uses the resultant index plane as the color-embedded grayscale version of the color image. The palette can be further embedded into the index plane and protected with a secret key. With the palette and the index plane on hand, one can easily reconstruct the color image.

Current VQ-based algorithms generally concern more on the quality of the reconstructed color image. The index plane produced after color quantization (i.e. the produced grayscale image) can deviate from the luminance plane of the original image very much. Besides, their reconstructed color images generally suffer from false contour and color shift due to its limited number of palette colors.

SE-based algorithms attempt to embed chromatic information into the luminance plane of the original color image by replacing the luminance content in the high frequency band with the downsampled chrominance content. The embedding can be done in the wavelet [5], DCT [6] or wavelet package transform [7,8,17] domain. Since the high frequency luminance content of the image is removed to make room for the chromatic information and the chrominance content is downsampled, the reconstructed color image is unavoidably blurred. Horiuchi et al. [8] argued to retain part of high frequency components to alleviate image blurring.

Another issue of SE-based algorithms is that, after being embedded into the luminance plane, the chromatic information is visible as textures in the color-embedded grayscale image. These textures form a highly-visible regular pattern in a smooth region as the chromatic information in a smooth region is spatially identical. Besides, current SE-based algorithms such as [5-8,17] do not protect color information in their reversible color-to-grayscale image conversion.

In this work, we aim at producing a high quality grayscale image with embedded chromatic information such that a high quality color image can be reconstructed based on the grayscale image itself when necessary. The quality of both the produced grayscale image and the reconstructed color image are equally emphasized. Our proposed solution is inspired by that digital halftoning techniques[20-24] can be used to shift the quantization noise to high frequency band such that the noise is hardly visible to the human eye[25].

A color-embedded grayscale image may be further converted with a halftoning algorithm into a binary halftone for being printed on paper. By taking this factor into account, there are attempts that combine an off-the-shelf RCGC algorithm and inverse halftoning to reconstruct a color image based on a binary halftone of a color-embedded grayscale image [15,16]. The issue that they address is different from the one concerned in this work. Their focus is on inverse halftoning instead of RCGC. In their works, halftoning is used to convert a color-embedded grayscale image into a binary halftone, rather than improve the performance of RCGC.

The rest of this paper is organized as follows. Section 2 presents our proposed RCGC algorithm in detail. Section 3 shows how the high frequency noise can be removed with a nonlinear low-pass filtering process. Section 4 provides some simulation results for performance evaluation and a conclusion is given in section 5.

2. Proposed Method

The proposed RCGC method is also based on color quantization. However, unlike those conventional algorithms[1-4,14], it makes use of the error diffusion technique to shift the quantization noise to high frequency region such that the noise can be less visible to human eyes. This noise shaping effect makes significant contributions to the performance of the proposed method. On one hand, it allows us to put more effort on improving the quality of the grayscale image as human can tolerate more high frequency noise in the reconstructed color image. On the other hand, besides the visual quality, the objective quality of the reconstructed color image can also be guaranteed as its high frequency quantization noise can be removed by low-pass filtering.

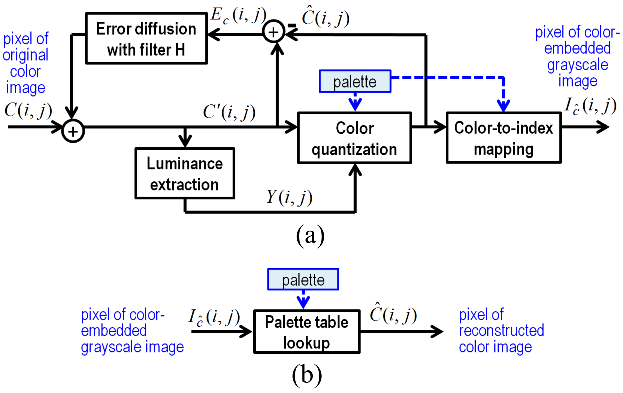
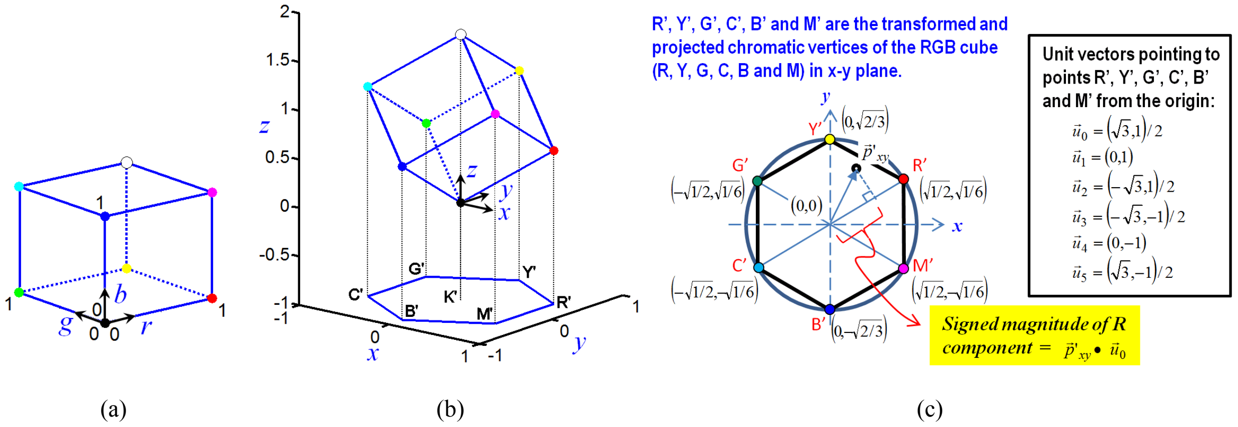


Figure 1 Block diagrams showing the flow of (a) color-to-grayscale conversion, and (b) grayscale-to-color mapping

Let C be the color image to be processed and $C(i,j)$ be the color of its pixel (i,j). Without loss of generality, $C(i,j)$ is represented as a vector in (r,g,b) format, where r, g and b are the intensity values of the red, green and blue components of the color and they are normalized to be in [0,1]. A palette is first derived based on image C to bear a property that the color index value of a palette color is highly correlated with the luminance value of the palette color. The details of its derivation are provided in Section 2.1.

The proposed RCGC method scans the image with serpentine scanning and processes the scanned pixels one by one as shown in Figure 1(a). $C'(i,j)$, $\hat{C}(i,j)$ and $E_c(i,j)$ are all intermediate processing results each of which represents a color. $Y(i,j)$ is the luminance value of color $C'(i,j)$. In this work, the luminance value of a color in (r,g,b) format is computed as $0.299r + 0.587g + 0.114b$ as in the commonly used RGB-to-YUV conversion. Based on $Y(i,j)$ and the pre-derived palette, color $C'(i,j)$ is quantized to $\hat{C}(i,j)$, a particular palette color whose corresponding color index value is denoted as $I_{\hat{c}}(i,j)$. $E_c(i,j) = C'(i,j) - \hat{C}(i,j)$ is actually the color quantization error. It will be diffused to the not-yet-processed neighboring pixels of pixel (i,j) in C. After processing all pixels, the index plane composed of $I_{\hat{c}}(i,j)$ for all (i,j) forms the resultant grayscale image.

The palette can be embedded into the grayscale image or stored separately with it. When necessary, pixel values of the grayscale image are used as indices to fetch



(a)
(b)
(c)
1
1
1
0
0 0
b
g
r
2
1.5
1
0.5
0
-0.5
-1
-1
z
y
x
C'
G'
K'
Y'
B'
M'
R'
0
x
1 -1
0
y
1
R', Y', G', C', B' and M' are the transformed and projected chromatic vertices of the RGB cube (R, Y, G, C, B and M) in x-y plane.
y
Y'
$(0,\sqrt{2/3})$
$\vec{p}'_{xy}$
G'
R'
$(-\sqrt{1/2},\sqrt{1/6})$
$(\sqrt{1/2},\sqrt{1/6})$
(0,0)
x
C'
M'
$(-\sqrt{1/2},-\sqrt{1/6})$
$(\sqrt{1/2},-\sqrt{1/6})$
B'
$(0,-\sqrt{2/3})$
Unit vectors pointing to points R', Y', G', C', B' and M' from the origin:
$\vec{u}_0=(\sqrt{3},1)/2$
$\vec{u}_1=(0,1)$
$\vec{u}_2=(-\sqrt{3},1)/2$
$\vec{u}_3=(-\sqrt{3},-1)/2$
$\vec{u}_4=(0,-1)$
$\vec{u}_5=(\sqrt{3},-1)/2$
Signed magnitude of R component = $\vec{p}'_{xy}\bullet\vec{u}_0$

Figure 2 Different domains for palette generation: (a) RGB color cube in *rgb* space, (b) transformed RGB color cube in *xyz* space and (c) the *x*-*y* plane used to derive a palette color

palette colors to reconstruct the color image as shown in Figure 1(b). An optional nonlinear low-pass filtering can be further applied to remove the high frequency noise in the reconstructed color image.

2.1 Palette development

To fit into our application, the developed palette should bear two properties. First, palette colors should be sorted according to their luminance values and indexed in a way that their luminance values are roughly proportional to their index values. This allows the index plane to appear as a grayscale image. Second, consecutive colors in the palette should form a three-dimensional enclosure in the color space to cover as many pixel colors (i.e. $C(i,j)$) that have the same luminance values as the involved palette colors as possible. Theoretically, with the halftoning technique, any specific color inside the enclosure can be rendered with the palette colors that form the enclosure as long as the region having the color to be rendered is large enough in image *C*.

There have been a lot of studies on how to optimize a color palette based on a given color image, and some of them also make an assumption that halftoning will be carried out in color quantization (e.g. [26-28]). However, none of them is developed to have the aforementioned properties because their palettes are optimized to provide a high quality color image and no reversible grayscale image is involved. A tailor-made palette is hence developed as follows to fit into our application.

Figure 2(a) shows the RGB color cube in *rgb* color space. All pixel colors that appear in color image *C* are in the RGB cube. To derive our palette, we stand the cube on the black vertex (0,0,0), with the white vertex (1,1,1) directly above it, as shown in Figure 2(b). This can be accomplished by rotating the cube about the *b*-axis by π/4 and then about the *r*-axis by θ=arcsin $(\sqrt{2/3})$, where θ is the angle between the achromatic axis (aligned with the line from (0,0,0) to (1,1,1) in the RGB cube shown in Figure 2(a)) and the *r*-, *g*- or *b*- axis.

After the rotations, a color in (*r*,*g*,*b*) format is mapped to (x,y,z) format as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) & -\sin\left(\frac{\pi}{4}\right) & 0 \\ \sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (1)$$

where z∈$[0,\sqrt{3}]$ provides the achromatic information of the color and (*x*,*y*) forms a vector that provides the chromatic information of the color. For reference purpose, we denoted a color in (*r*,*g*,*b*) format as $\vec{p}$ and its representation in (*x*,*y*,*z*) format as $\vec{p}'=T\vec{p}$, where *T* is the transformation matrix given in eqn. (1).

Without loss of generality, we assume that a palette is consisted of 256 colors and they are indexed from 0 to 255. To determine the k^{th} palette color, we collect all pixel colors the luminance values of which are in the range of $B_l(k)=\max\left(0,\frac{k-2}{255}\right)$ and $B_u(k)=\min\left(1,\frac{k+3}{255}\right)$, project them onto the two-dimensional *x*-*y* plane as shown in Figure 2(b), compute their signed magnitudes of a selected primary color component, and find out the one which gets the maximum. In our algorithm, six primary color components including red(R), yellow(Y), green(G), cyan(C), blue(B) and magenta(M) are considered. Figure 2(c) provides an example that shows how to compute the signed magnitude of a pixel color's primary red component after it is transformed and projected onto the *x*-*y* plane, where $\vec{p}'_{xy}=(x,y)$ is a vector formed by the components *x* and *y* of $\vec{p}'=T\vec{p}$.

The pixel color having the maximum signed magnitude of a selected primary color component is the outermost pixel color in $\Omega_k=\{\,\vec{p}\mid\vec{p}\in$ color image C and $B_l(k)\le$ luminance value of $\vec{p}\le B_u(k)\,\}$ in terms of the distance from the *z*-axis along a specific direction. To make every six contiguous palette colors form an enclosure to enclose more pixel colors in Ω_k , when picking the primary color component for deriving the k^{th} palette color, we pick the six chromatic vertices (i.e. R, Y, G, C, B and M) in the RGB cube in turns as k increases. In formulation, the k^{th} palette color $\vec{c}_k$ is given as

$$\vec{c}_k = \arg\max_{\vec{p}\in\Omega_k} \vec{p}\,'_{xy} \bullet \vec{u}_{\operatorname{mod}(k,6)} \qquad \text{for } k=0,1\ldots255 \qquad (2)$$

where $\bullet$ is the dot product operator, mod(k,6) means k modulo 6, and $\vec{u}_i$ for $i = 0,1 \ldots 5$ are the unit direction vectors shown in Figure 2(c).

With this arrangement, the k^{th} palette color is the outermost color from the origin of the *x*-*y* plane along a specific direction among all image pixel colors whose luminance values belong to [$B_l(k)$, $B_u(k)$] as shown in Figure 3. Besides, every 6 consecutive palette colors form an enclosure that contains almost all image pixel colors that have the same luminance levels with the palette colors. The enclosure may not be able to enclose all image pixel colors, but those outside the enclosure are very close to the enclosure boundary. That means, for any given image pixel color $\vec{p}$, we can use 6 palette colors to render it and their luminance errors from $\vec{p}$ are bounded in [-2/255, 3/255].

This capability cannot be achieved with conventional color palettes even though error diffusion is used to render $\vec{p}$ because there may not be palette colors having similar luminance levels and there can be a lot of image pixel colors far outside the enclosure formed with the palette colors having similar luminance levels (e.g. the enclosure in red) as shown in Figure 3. Figure 4 shows the consequences of using two palettes of different properties to reconstruct a color image. As shown in Figure 4(b), even though error diffusion is exploited, there can be serious color shift in the reconstructed color image if the used palette does not have the aforementioned properties.

In general, we can design a palette that uses d consecutive palette colors to form an enclosure to enclose image pixel colors whose luminance levels are in [max(0, floor(k-$d/2$+1)/255),min(1,floor(k+$d/2$)/255)]. It can be achieved by using d different unit direction vectors $\vec{u}_i$ alternately when determining the k^{th} palette color with eqn.(2). For comparison study, we developed three palettes for Figure 4(a) under three different settings (d=3, 6 and 9). Figure 5 shows the enclosures formed with the three palettes when k=100. One can see that a triangle can leave many pixel colors outside while using a dodecagon instead of a hexagon helps little to enclose more pixel colors. In view of this, we select d=6 in our realization.

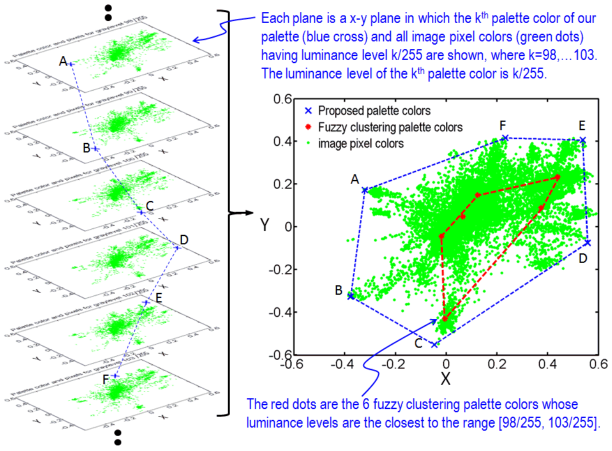
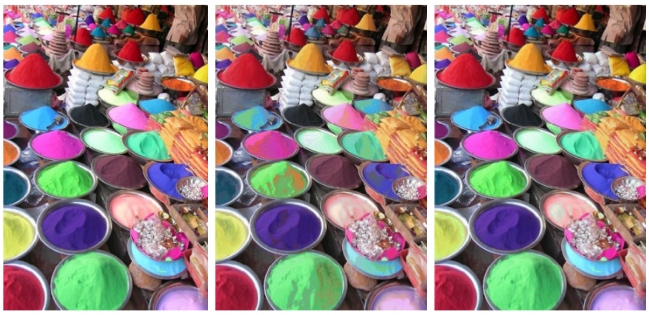
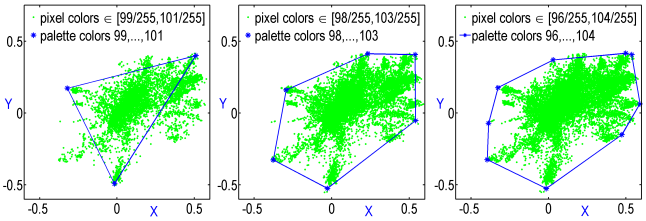


Figure 3. An example showing how 6 consecutive palette colors in our palette form an enclosure to cover most image pixel colors having similar luminance levels. The palette is developed based on the image shown in Figure 4(a).



(a) (b) (c)

Figure 4. Results of using different palettes in the proposed RCGC algorithm to reconstruct a color image: (a) original, (b) a palette obtained with fuzzy clustering [18] and (c) the proposed palette.



(a) (b) (c)

Figure 5. Enclosures formed with d palette colors when the proposed palette is developed under different settings: (a) d=3, (b) d=6, and (c) d=12. The data is obtained for k=100 and the palettes are developed for the image shown in Figure 4(a).

Though the palette generation process seems to be complicated, it can be realized easily as the involved projection and transformation are simple and the chromatic vertices of the RGB cube only contain component values 0 and 1. After simplification, eqn. (2) can be rewritten as

$$\vec{c}_k = \arg\max_{\vec{p}\in\Omega_k} \left(\vec{v}_{\operatorname{mod}(k,6)} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} r_p \\ g_p \\ b_p \end{bmatrix} \right)$$

$$\text{for } k=0,1\ldots255 \qquad (3)$$

where $\vec{v}_0=(1,0,0)$, $\vec{v}_1=(1,1,0)$, $\vec{v}_2=(0,1,0)$, $\vec{v}_3=(0,1,1)$, $\vec{v}_4=(0,0,1)$ $\vec{v}_5=(1,0,1)$ and (r_p,g_p,b_p) is pixel color $\vec{p}$ in (r,g,b) format.

In case Ω_k is empty for a particular k, palette color $\vec{c}_k$ for that index value k is determined as

$$\vec{c}_k = \arg\max_{\vec{p}\in\Omega''_k} \vec{p}'_{xy} \bullet \vec{u}_{\mathrm{mod}(k,6)} \qquad \text{if } \Omega_k \text{ is empty} \qquad (4)$$

where $\Omega''_k=\{\ \vec{p}\mid\vec{p}\in$ surface of the RGB color cube and $B_l(k)\le$ luminance value of $\vec{p}\le B_u(k)\ \}$. It is image independent and hence can be precomputed.

As a final remark, we note that the proposed palette generation algorithm is not an iterative algorithm. The whole palette can be determined by scanning the image once and hence it is able to support real-time applications.

2.2 Color quantization

Constrained color quantization is performed in the proposed RCGC algorithm. For any input color $C'(i,j)$, its luminance value $Y(i,j)$ is computed and then used to confine a subset of the palette colors to one of which we can quantize $C'(i,j)$. In particular, the selected subset of palette colors is given as

$$\Gamma=\{\vec{c}_k \mid -5\le k-round(255\times Y(i,j))\le 6 \ \ and\ \ 0\le k\le 255\} \qquad (5)$$

As CIEL*a*b* color space is device independent and perceptually uniform, we select the Euclidean distance in CIEL*a*b* space as the quantization criterion. In formulation, we have

$$\hat{C}(i,j)=\arg\min_{\vec{c}_k\in\Gamma}\left\|T_{CIELAB}(C'(i,j))-T_{CIELAB}(\vec{c}_k(i,j))\right\| \qquad (6)$$

where $T_{CIELAB}(\bullet)$ denotes the transform from *rgb* space to CIEL*a*b* space. The index value of $\hat{C}(i,j)$ is $I_{\hat{C}}(i,j)$.

Unlike conventional color quantization algorithms that search for the best color from 256 palette colors, the proposed algorithm only search at most 12 palette colors. It significantly reduces the realization effort and bounds the maximum luminance distortion of a pixel.

2.3 Error diffusion

Error diffusion is used to shape the color quantization noise such that most of the noise energy is in the high frequency band. When processing pixel (i,j), the color errors introduced to its processed neighbors are taken into account and its color is adjusted as

$$C'(i,j)=C(i,j)+\sum_{(m,n)\in\Theta}E_c(i-m,j-n)h(m,n) \qquad (7)$$

where $\Theta=\{(m,n)\ |m=0,1 \text{ and } n=-1,0,1\}$ is the support of the diffusion filter H and $h(m,n)$ is the $(m,n)^{th}$ coefficient of filter H for $(m,n)\in\Theta$. In our realization, we adopt the diffusion filter that is commonly used in binary halftoning [20]. Specifically, it is defined as

$$H=\frac{1}{16}\begin{bmatrix}0 & * & 7\\ 3 & 5 & 1\end{bmatrix} \qquad (8)$$

where * marks the pixel being processed and it corresponds to filter coefficient $h(0,0)$. The diffusion is performed in each channel separately.

2.4 Embedding the palette

For applications where color information needed to

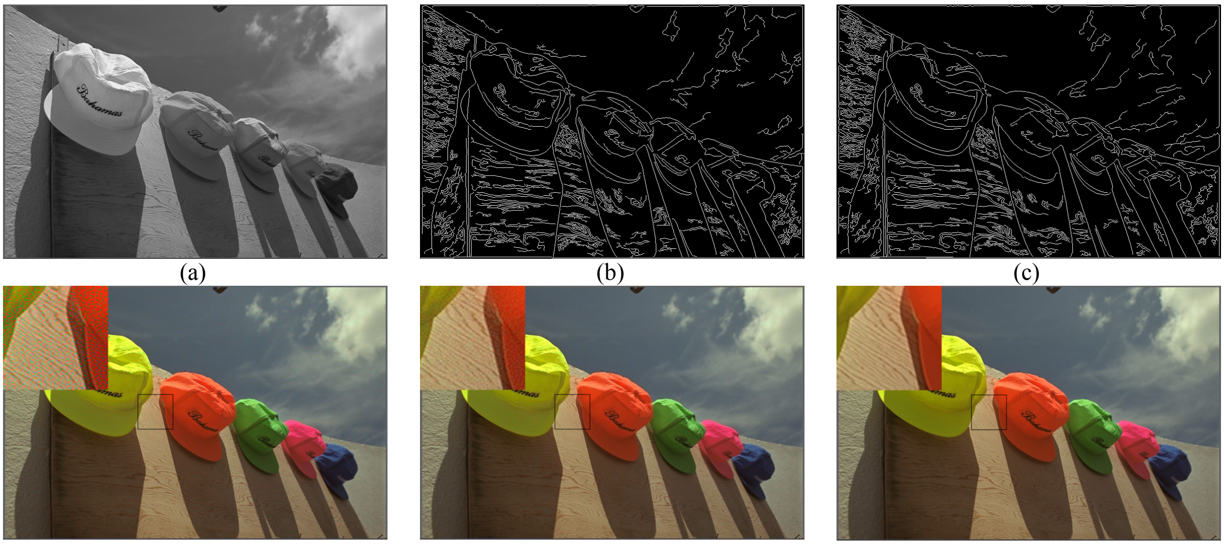

(a) (b) (c)
(d) (e) (f)

Figure 6 Recovering color image from our color-embedded grayscale image: (a) color-embedded grayscale image $I_{\hat{C}}$; (b) edge map obtained with Fig. 6(a); (c) edge map obtained with the original image C; (d) color image obtained before non-linear low-pass filtering, $\hat{C}$; (e) color image obtained after nonlinear low-pass filtering, $\tilde{C}$; (f) original color image C.

be protected, the palette can be embedded into the grayscale image (i.e. the index plane $I_{\hat{C}}$) with encryption. It is also helpful to reduce the overhead. The approach presented in [4] is modified in the proposed algorithm to achieve this goal. In particular, the whole palette is rearranged to form a bit sequence of length L. The index plane is then partitioned into L non-overlapped blocks. From each block, a pixel is selected to carry a bit from the bit sequence. A pseudo-random number generator (PRNG) with a secret key is applied to select the pixels into which the palette information is embedded.

The palette-embedding process is taken into account during the color-to-grayscale conversion to minimize its distortion to the conversion output. For pixels which are not selected to carry the palette information, they are processed as mentioned before. For those are selected, they are color-quantized differently as follows:

$$\hat{C}(i,j) = \underset{\vec{c}_k \in \Gamma_s}{\arg\min} \left\| T_{CIELAB}(C'(i,j)) - T_{CIELAB}(\vec{c}_k(i,j)) \right\| \quad (9)$$

where $s \in \{0,1\}$ is the value of the bit embedded and

$$\Gamma_s = \{\vec{c}_k \mid \mathrm{mod}(k,2) = s \text{ and } \vec{c}_k \in \Gamma\} \quad \text{for } s=0,1 \quad (10)$$

As the error introduced by palette-embedding will be diffused with the quantization error to the neighboring pixels, its spectrum will also be shaped and its impact on the quality of the output images can be minimized.

Palette extraction is easy with the secret key. The secret key allows us to locate the pixels that carry the embedded information and their least significant bits form the bit sequence for us to reconstruct the palette.

The basic idea of the method used here stems from the ±1 LSB embedding algorithm [29]. This algorithm is widely applied in image steganography. According to the investigation in [4], this method is simple and efficient enough to protect the color information from being acquired by unauthorized people without a security key. Since we are not doing watermarking, whether the embedded information can stand for attacks is not the issue.

3. Halftone Artifacts Suppression

Figures 6(a) and 6(d) show, respectively, $I_{\hat{C}}$, the color-embedded grayscale image produced with the proposed algorithm, and $\hat{C}$, the color image directly reconstructed by using $I_{\hat{C}}$ as the index plane and the palette embedded in $I_{\hat{C}}$. As shown in Figure 6(d), there is no artifact such as color shift and false contour in $\hat{C}$, and its visual quality is already good at a reasonable viewing distance. The error diffusion step shifts the quantization noise to high frequency band such that the noise can be removed by the low-pass filtering effect of human eyes. However, the high frequency noise can still be visible when we get too close to view the image. A descreening scheme is hence applied to alleviate this problem.

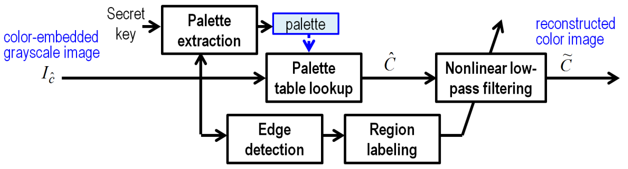


Figure 7 Block diagram of color image reconstruction with halftone artifacts suppression

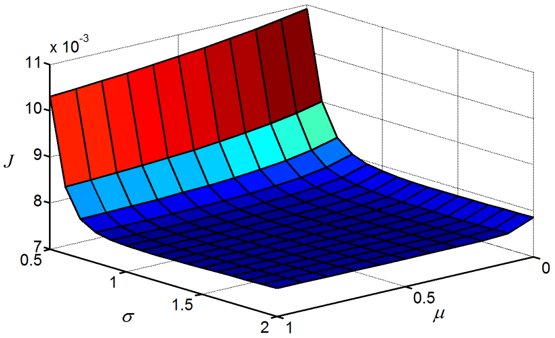


Figure 8 The plot of *J* vs. σ and μ

At first glance, bilateral filtering [30] seems to be good for handling the situation as it is designed to preserve sharp edges and remove noise simultaneously. However, our simulation results show that it does not work properly in this scenario. After error diffusion, the grayscale of a pixel in the color-embedded grayscale image can fluctuate a bit from the original. As it is used as an index to determine the color of the pixel, a minor grayscale difference among adjacent pixels can lead to a significant chrominance difference. Pixels on the same side of an edge may not have similar chroma values and hence bilateral filtering does not work properly.

In our solution, a simple nonlinear low-pass filter is used to remove the noise as follows:

Step 1: Apply Canny edge detector [31] to the color-embedded grayscale image to get a binary edge map of the image.

Step 2: Label different regions according to the edge map.

Step 3: Derive a spatial-variant low-pass filter $\mathbf{F}_{(i,j)}$ for pixel (*i*,*j*) as

$$\mathbf{F}_{(i,j)}(m,n) = \begin{cases} A \cdot \exp\left(-(m^2+n^2)/\sigma^2\right) \\ \quad \text{if pixels } (i,j) \text{ and } (i+m,j+n) \\ \quad \text{are in same region;} \\ A \cdot \mu \cdot \exp\left(-(m^2+n^2)/\sigma^2\right) \\ \quad \text{else} \end{cases}$$

$$\text{for } -4 \le m,n \le 4 \quad (11)$$

where $\mathbf{F}_{(i,j)}(m,n)$ is the $(m,n)^{\text{th}}$ filter coefficient of filter $\mathbf{F}_{(i,j)}$, σ controls the cut-off frequency of the filter, A is a normalization factor that makes the weights in the kernel sum to one, and $\mu\in[0,1]$ is a controlling parameter that controls the contribution of the pixels from different regions. Specifically, $\mu=1$ means that the edge information is ignored and the filtering is linear, and $\mu=0$ means that pixels not in the same region with the pixel being processed are ignored.

Step 4: Apply filter $\mathbf{F}_{(i,j)}$ to the chromatic channels of the reconstructed color image $\hat{C}$ to remove the noise as follows.

$$a^*(i,j)=\sum_{|m|,|n|\le 4}\mathbf{F}_{(i,j)}(m,n)a^*(i+m,j+n)$$

$$b^*(i,j)=\sum_{|m|,|n|\le 4}\mathbf{F}_{(i,j)}(m,n)b^*(i+m,j+n)$$

$$\text{for all } (i,j) \quad (12)$$

where $a^*(i,j)$ and $b^*(i,j)$ are, respectively, the $(i,j)^{\text{th}}$ pixels of channels a^* and b^* of color image $\hat{C}$ in CIEL*a*b* color space.

The luminance channel is not filtered so as to preserve the texture information of the image. While Figure 1(b) shows the reversion of our color-to-grayscale conversion without halftone artifacts suppression, Figure 7 shows a complete picture of how our proposed algorithm reconstructs the color image by including both palette extraction and nonlinear low-pass filtering in the operation flow. The final reconstructed color image is $\tilde{C}$.

Parameters σ and μ were trained in our simulation based on a set of testing images (24 images from Kodak set [13]) to minimize objective function $J=\| C-\tilde{C}\|^2$, where C and $\tilde{C}$ are, respectively, the original and the final reconstructed color images. Figure 8 shows how J changes with σ and μ. It is found that J reaches its minimum at $\sigma=1.22$ and $\mu=0.7$. All subsequent simulation results reported in this paper are obtained based on these settings. However, as shown in Figure 8, J is actually insensitive to σ and μ when $\sigma\in[0.8,2.0]$ and $\mu\in[0.2,1.0]$. Hence, some other settings of σ and μ can also be used to achieve the same performance.

Figure 6 illustrates step-by-step how $\tilde{C}$ is reconstructed with $I_{\hat{C}}$. Figure 6(b) is the edge map acquired from the index image shown in Figure 6(a). As a reference, Figure 6(c) shows an edge map obtained with the original image. One can see that most of the edges can be successfully extracted based on $I_{\hat{C}}$. It is hence reliable to utilize the edge information in nonlinear low-pass filtering. Figures 6(d), (e) and (f) respectively, show $\hat{C}$, $\tilde{C}$ and the original image C. On their left upper corners, the enlarged versions of the regions enclosed in black boxes are displayed for better inspection. One can see that the high frequency impulse noise in $\hat{C}$ can be efficiently removed after filtering, while the texture information in the original image is preserved.

Figure 9 Test image set 2

4. Simulation results

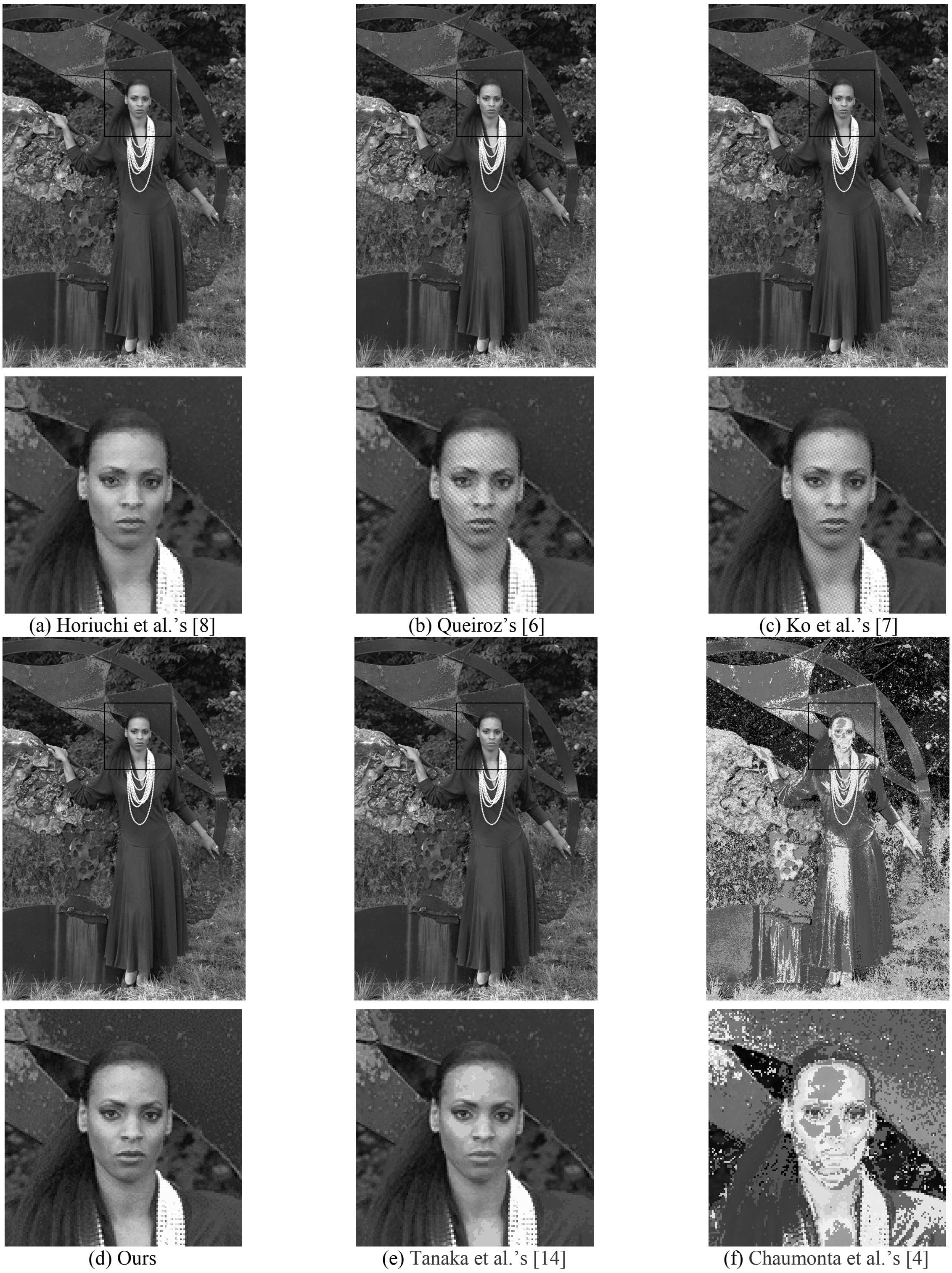
Simulations were carried out to evaluate the performance of the proposed RCGC algorithm. In our simulations, some other state-of-art conversion algorithms including Queiroz's [6], Ko et al.'s [7], Horiuchi et al.'s [8], Tanaka et al.'s [14] and Chaumont et al.'s [4] were also evaluated for comparison. Among them, the first three are SE-based algorithms while the others are VQ-based algorithms. When realizing Queiroz's [6], a 4×4 DCT was applied and the highest six frequency bands were used to embed the color information. When realizing Ko et al.'s [7], a four-level wavelet packet transform was implemented and the two subbands with minimum amount of information were used to embed color. Accordingly, the two chroma planes were subsampled by four in each direction. In the realization of [14], a one-level discrete wavelet transform was applied and the chroma information was downsampled by 16 in each direction. In all evaluated SE-based algorithms, bilinear interpolation was exploited to resize the chrominance planes when reconstructing a color image. When realizing Chaumont et al.'s[4], the palette size is image-dependent and determined by optimizing a cost function as suggested in [4]. As for the other VQ-based algorithms, the palette size is fixed to be 256.

Two sets of testing color images are used. One is the Kodak set that includes 24 color images of size 768×512 or 512×768[13]. The other one includes the images shown in Figure 9 and each of them is of size 512×512. Figures 10 and 11 show the simulation results for test image *'woman'* in Kodak set [13]. For comparison, Figure 12 shows the original image as the ground truth.

Image blurring is a common issue of SE-based algorithms [6-8] because some texture information is discarded to make room for embedding the color information. This problem can be alleviated to some extent if the chrominance planes are compressed with

JPEG coding before being embedded. However, subband embedding also gives rise to another issue that serious pattern noise can appear in the color-embedded grayscale images as shown in Figures 10(b) and 10(c). Comparatively, Figure 10(a) shows less pattern noise but it is still visible in the neck and the forehead. When inspecting their reconstructed color images, one can see color shift in some local regions (e.g. pearl necklaces in Figures 11(a) and 11(b), and the skin and lip in Figure 11(c)). Figure 11(b) is obviously blurred as compared with Figure 12(a). Figure 11(a) actually does not preserve original texture details properly. One can see in Figure 11(a) that the eye and the ear on the left are seriously distorted by local high frequency noise.

Theoretically, VQ-based algorithms [4,14] can achieve good mean square error (MSE) performance since their palettes are optimized to minimize the quantization error. However, without the help of noise



(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

(c) Ko et al.'s [7]

(d) Ours

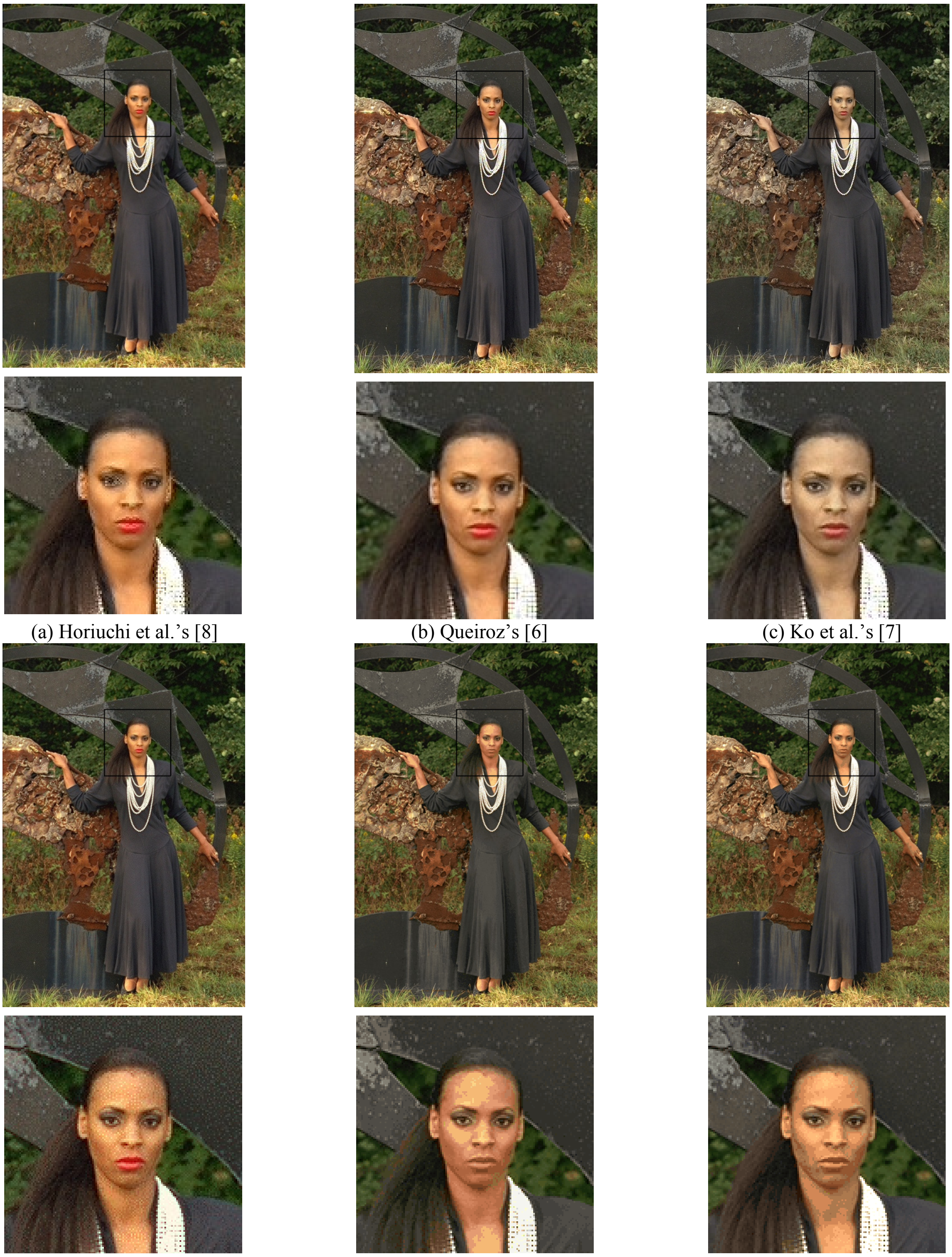
(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

Figure 10. Color-embedded grayscale images obtained with various algorithms (Test image: Woman)

shaping, the limited number of palette colors generally results in false contours and color shift. As shown in Figures 11(e) and 11(f), one can easily see the color shift on the lip and the false contours on the face.

As shown in Figure 13, in contrast to the palettes associated with Figures 11(e) and 11(f), the palette colors of the palette associated with Figure 11(d) are more scattered in the color space. In fact, our palette generation method picks pixel colors of similar luminance levels and then selects the outermost ones in the *x*-*y* plane as palette colors. As a consequence, most pixel colors in the image can be enclosed by some palette colors of similar luminance levels. Figure 14 shows the distributions of all our palette colors in the *x*-*y* plane for different testing images. By exploiting the error diffusion technique, the proposed RCGC algorithm is able to render much more than 256 colors and preserve the spatial texture of an image. As shown in Figure 11(d), our reconstructed color


(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

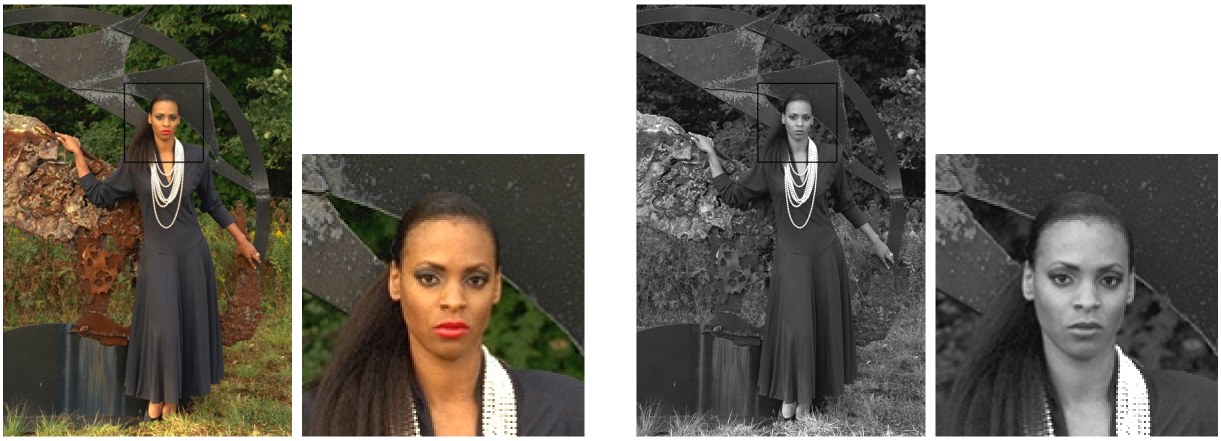
(c) Ko et al.'s [7]

(d) Ours

(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

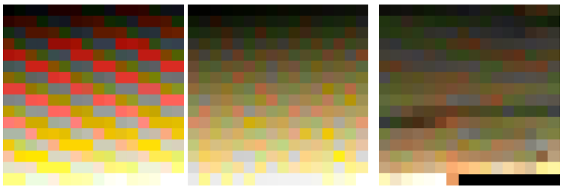
Figure 11. Recovered color images obtained with various algorithms (Test image: Woman)



(a) Original

(b) Luminance plane of original

Figure 12. Ground truth for Figures 10 and 11 (Test image: Woman)



(a)

(b)

(c)

Figure 13. Palettes associated with (a) Figure 11(d), (b) Figure 11(e) and (c) Figure 11(f). Their sizes are 256, 256 and 247 respectively.

image is sharp and there is no color shift or saturation loss. The noise carried in the reconstructed color image is high frequency noise that is invisible from a reasonable distance.

Chaumont et al.'s algorithm [4] does not take the construction of the grayscale image into account when optimizing the palette and hence the produced grayscale image could be very different from the ground truth as shown in Figure 10(f). Tanaka et al.'s algorithm [14] can produce a better grayscale image by optimizing the palette under a lightness constraint but false contour can still be easily observed. In contrast, our color-embedded grayscale image shown in Figure 10(d) is very close to the ground truth, which increases the security of the hidden color information as a visually normal grayscale image does not alert people.

Figures 15 and 16 provide the simulation results of another testing image for inspection and similar observations can be obtained. One can find severe pattern noise in Figures 15(a)-(c), false contours in Figures 15(e)-(f), color noise or blurring at the edges in Figures 16(a)-(c), and color shift in Figures 16(e)-(f).

The performance of the evaluated algorithms is also compared in terms of some other objective metrics such as *HVS-PSNR* [32], *SSIM* [33], s-CIELAB distance $\Delta E_{s-CIELAB}$ [34,35], *HVS-PSNR*$_{Color}$, *CSSIM* [36] and *SSF* [37]. Among them, the first two are for evaluating the quality of a color-embedded grayscale image while the others are for evaluating the quality of a reconstructed color image. *HVS-PSNR*$_{Color}$ is an extension of *HVS-PSNR* [32] and it is calculated as

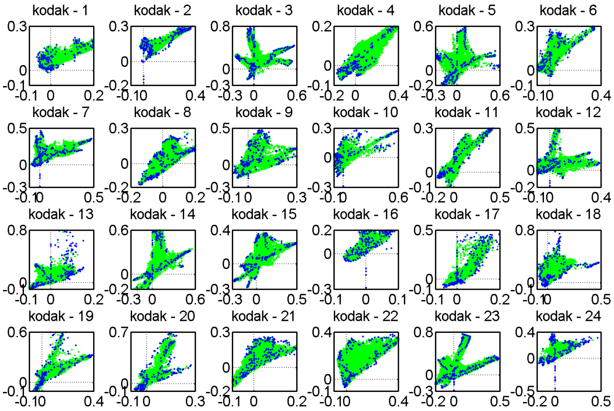


Figure 14 Distributions of image pixel colors and our palette colors in *x-y* space for different testing images in Kodak image set. Green dots and blue dots denote the image pixel colors and the palette colors respectively.

$$HVS-PSNR_{Color}=10\log_{10}\frac{3MN}{\|F_G\otimes(C-\tilde{C})\|^2} \qquad (13)$$

where F_G specifies a Gaussian filter that simulates the low-pass property of human visual system [38], and $\otimes$ symbolizes the 2-D convolution operation.

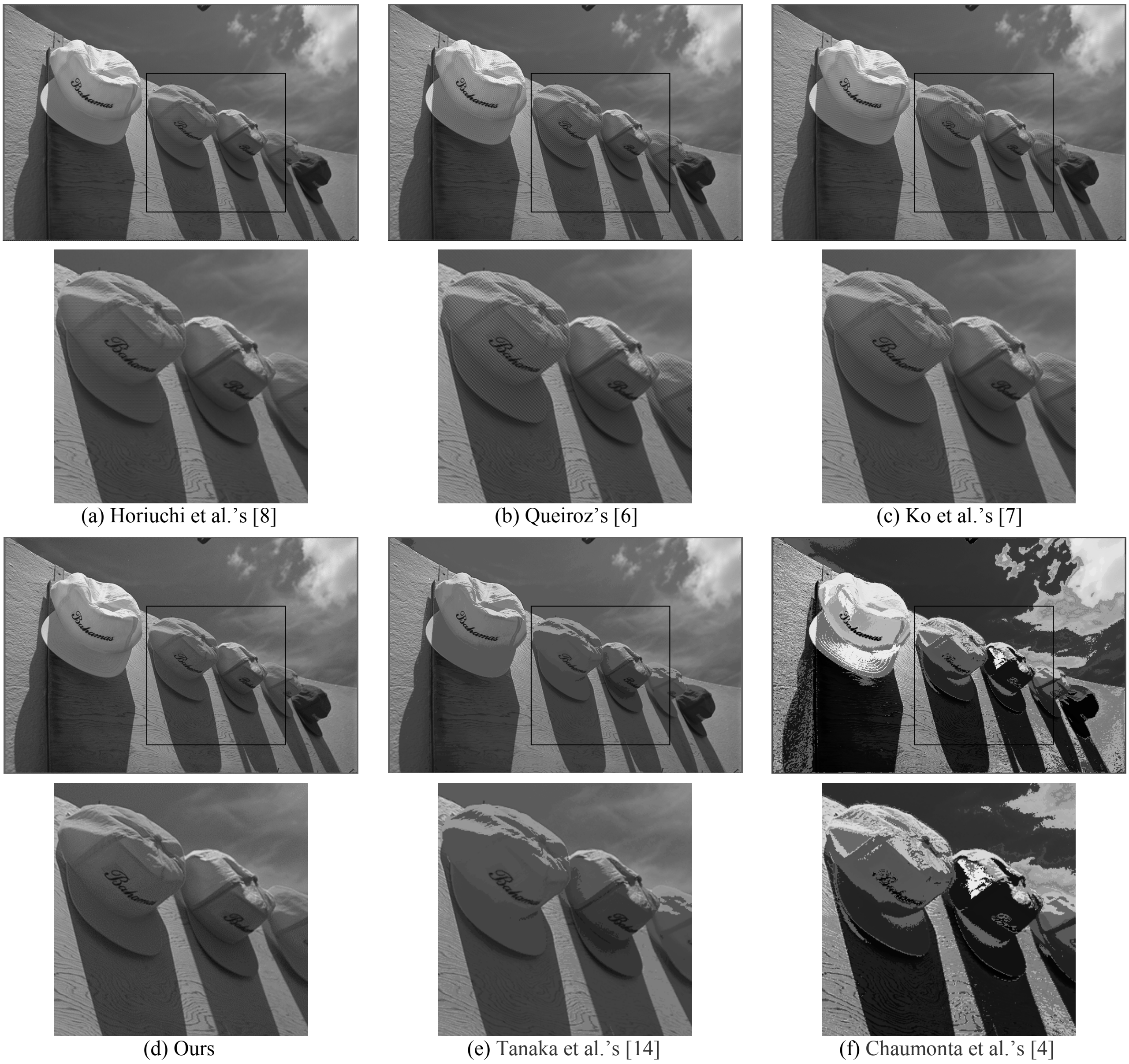
For assessments that involve human visual system model (*HVS-PSNR*, *HVS-PSNR*$_{Color}$, $\Delta E_{s-CIELAB}$ and *CSSIM*), we consider the cases where the resolutions are, respectively, 100 and 150 dpi with a fixed viewing distance of 20 inches.

The performances shown in Tables 1 and 2 are, respectively, based on the simulation results obtained with testing image sets 1 and 2. Note that the parameters μ and σ for the nonlinear low-pass filtering module used in our algorithm is trained with testing image set 1 only. The data shown in table 2 hence reflect whether the parameters can also work properly with images that do not involved in the training. The findings based on the data reported in both Tables are consistent, so we may consider that the obtained parameters μ and σ are universally valid.

As far as the quality of the color-embedded grayscale image is concerned, the proposed algorithm provides the highest *PSNR* performance. Though there is severe pattern noise in the outputs of SE-based algorithms, Ko et. al's algorithm performs well in terms of *HVS-PSNR*. It is because the frequency of its pattern noise is high enough to make the noise removable by the lowpass filter used to simulate our HVS when computing *HVS-PSNR.*

As for the recovered color images, our proposed algorithm gets leading scores in many assessments except *CPSNR*. The palettes used in [4] and [14] are designed to minimize the mean square error between palette colors and pixel colors and hence they can achieve better *CPSNR* scores. However, by considering that there are false contours and color shift in their reconstructed images as shown in Figures 11 and 16 and that the quality of their color embedded grayscale image is very poor as shown in Figures 10 and 15, their performance is actually inferior to ours.

As a final remark, we note that Figures 10(d) and 15(d) take 0.0156 bpp to carry their embedded palettes that carry the chromatic information of Figures 11(d) and 16(d) respectively. In fact, this figure is inversely proportional to the image size in terms of number of pixels. When one uses the naive approach in which the chromatic information is compressed directly and then embedded as binary data into the grayscale image with a general data hiding algorithm, the chromatic information has to be compressed at a compression ratio of 1026. Like other RCGC algorithms, the proposed algorithm does not take JPEG coding into account during its development and hence the embedded information is fragile to JPEG coding.


(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

(c) Ko et al.'s [7]

(d) Ours

(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

Figure 15. Color-embedded grayscale images obtained with various algorithms (Test image: Caps)

5. Conclusions

This paper presents a novel VQ-based RCGC scheme that emphasizes the quality of both the color-embedded grayscale image and the reconstructed color image. The improved quality of the images is mainly achieved by developing a color palette that fits into the application and exploiting error diffusion to shape the quantization noise to high frequency band. Simulation results show that the performance of the proposed algorithm is superior to conventional RCGC algorithms subjectively and objectively in terms of different metrics.

Acknowledgement

The work described in this paper is supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project No: PolyU5120/13E) and the Hong Kong Polytechnic University.

References

[1] M. Chaumont and W. Puech, "A grey-level image embedding its color palette," Proc. of ICIP, pp.389-392, 2007.

[2] M. Chaumont and W. Puech, "A fast and efficient method to protect color images," Proc. SPIE 6508, Visual Communications and Image Processing 2007, 65081T, 2007.

[3] M. Chaumont and W. Puech, "Protecting the color information by hiding it," Recent Advances in Signal Processing, ISBN 978-953-7619-41-1, 2009.

[4] M. Chaumont, W. Puechb and C. Lahanier, "Securing Color Information of an Image by concealing the color palette," Journal of Systems and Software, vol.86, no.3, pp.809-825, 2013.

[5] R.L. de Queiroz and K.M. Braun, "Color to gray

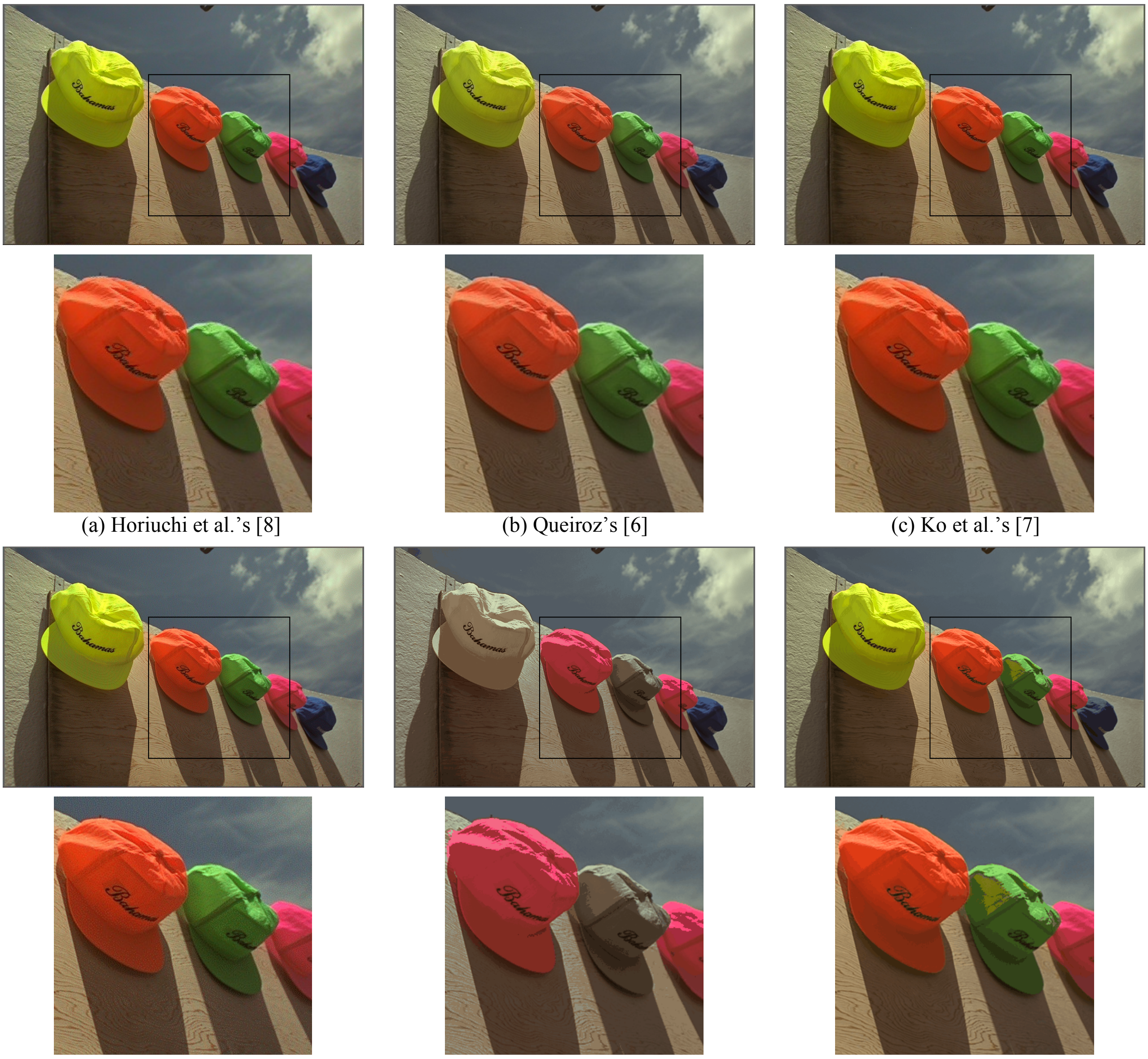

(a) Horiuchi et al.'s [8]
(b) Queiroz's [6]
(c) Ko et al.'s [7]
(d) Ours
(e) Tanaka et al.'s [14]
(f) Chaumonta et al.'s [4]

Figure 16. Recovered color images obtained with various algorithms (Test image: Caps)

Method	*Color-embedded Grayscale Image*				*Recovered Color Image*							
	PSNR (dB)	*SSIM*	*dpi = 100*	*dpi = 150*	*CPSNR* (dB)	*SSF*	*dpi = 100*			*dpi = 150*		
			HVS-PSNR (dB)	*HVS-PSNR (dB)*			*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*	*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*
Queiroz [6]	31.64	0.9760	40.92	43.85	31.84	**0.9998**	36.78	0.877	1.754	38.09	0.889	1.630
Ko et al. [7]	32.82	0.9896	**42.88**	**46.67**	32.13	0.9995	36.05	0.838	3.433	37.01	0.848	3.272
Horiuchi et al. [8]	33.20	**0.9997**	39.19	43.66	29.53	0.9997	33.69	0.927	1.488	36.02	0.952	1.214
Tanaka et al. [14]	28.07	0.9866	29.79	31.19	33.07	0.9986	34.78	0.874	2.547	35.51	0.879	2.435
Chaumont et al. [4]	16.72	0.6432	17.92	18.47	**37.79**	**0.9998**	**39.94**	0.962	1.017	40.99	0.968	0.921
Proposed	**35.87**	0.9883	41.54	44.01	32.71	**0.9998**	38.04	**0.963**	**0.790**	**41.24**	**0.968**	**0.576**

Table 1 Performance of RCGC algorithms for Kodak image set [13]

Method	*Color-embedded Grayscale Image*				*Recovered Color Image*							
	PSNR (dB)	*SSIM*	*dpi = 100*	*dpi = 150*	*CPSNR* (dB)	*SSF*	*dpi = 100*			*dpi = 150*		
			HVS-PSNR (dB)	*HVS-PSNR (dB)*			*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*	*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*
Queiroz [6]	29.89	0.9615	39.59	43.12	28.68	0.9987	30.32	0.716	3.111	31.22	0.747	2.845
Ko et al. [7]	30.26	0.9799	41.18	**46.06**	29.88	0.9981	31.42	0.823	2.509	32.18	0.856	2.192
Horiuchi et al. [8]	**35.95**	**0.9998**	40.87	43.24	29.06	0.9985	30.97	0.841	2.323	31.99	0.877	1.961
Tanaka et al. [14]	31.31	0.9743	32.75	33.56	33.24	0.9972	34.85	0.874	1.916	37.22	0.886	1.776
Chaumont et al. [4]	16.03	0.6271	16.96	17.35	**34.41**	0.9987	**36.85**	0.919	1.481	38.32	0.929	1.335
Proposed	**35.95**	0.9861	**41.33**	43.51	31.47	**0.9995**	36.45	**0.956**	**0.966**	**39.52**	**0.964**	**0.727**

Table 2 Performance of RCGC algorithms for test image set 2

and back: color embedding into textured gray images," IEEE Trans. Image Process., vol.15, pp. 1464–1470, 2006.

[6] R.L. de Queiroz, "Reversible color-to-gray mapping using subband domain texturization," Pattern Recogn. Lett., vol.31, pp.269–276, 2010.

[7] K.W. Ko, O.S. Kwon, C.H. Son and Y.H. Ha, "Color embedding and recovery based on wavelet packet transform," Journal of Imaging Science and Technology, vol.52, pp.010501, 2008.

[8] T. Horiuchi, F. Nohara and S. Tominaga, "Accurate reversible color-to-gray mapping algorithm without distortion conditions," Pattern Recogn. Lett., vol.31, pp.2405–2414, 2010.

[9] W. Zhang, X. Hu, X. Li and N. Yu, "Recursive histogram modification: establishing equivalency between reversible data hiding and lossless data compression," IEEE Trans. Image Process. vol.22, pp.2775–2785, 2013.

[10] B. Ou, X. Li, Y. Zhao, R. Ni, Y.Q. Shi, "Pairwise prediction-error expansion for efficient reversible data hiding," IEEE Trans. Image Process. Vol.22, pp.5010–5021, 2013

[11] I.C. Dragoi, D. Coltuc, "Adaptive pairing reversible watermarking," IEEE Trans. Image Process. Vol.25, pp.2420–2422, 2016.

[12] C. Poynton, Digital video and HDTV algorithms and interfaces, (San Francisco: Morgan Kaufmann, 2003)

[13] "Kodak true color image suite," [Online]. Available: http://r0k.us/graphics/kodak/

[14] G. Tanaka, N. Suetake and E. Uchino, "Invertible color-to-monochrome transformation based on clustering with lightness constraint," IEEE conference on SMC, pp.2151-2154, 2010.

[15] C.H. Son and H. Choo, "Color recovery of black-and-white halftoned images via categorized color-embedding look-up tables," Digital Signal Processing, vol.28, pp.93-105, 2014.

[16] C.H. Son, K. Lee and H. Choo, "Inverse color to black-and-white halftone conversion via dictionary learning and color mapping," Information Sciences, vol. 299, pp. 1-19, 2015.

[17] K.W. Ko, D.C. Kim, W.Y. Kyung and Y.H. Ha, "Color embedding and recovery using wavelet packet transform with pseudorandomized saturation code," Journal of Imaging Science and Technology, vol.55, 030501.1-10, 2011.

[18] Bezdek, C. James, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," Computers & Geosciences, 10(2), pp.191-203, 1984.

[19] J.A. Hartigan and M.A. Wong, "Algorithm AS 136: A K-means clustering algorithm," Journal of the Royal Statistical Society, 28(1), pp.100–108, 1979.

[20] R.W.Floyd and L.Steinberg, "An adaptive algorithm for spatial gray scale," Proc. SID 75 Digital: Society for Information Display, pp.36-37, 1975.

[21] P. Li and J. P. Allebach, "Tone dependent error diffusion," IEEE Trans. Image Process., vol.13, pp.201-215, 2004.

[22] Y.H. Fung and Y.H. Chan, "Tone-dependent error diffusion based on an updated blue-noise model," J.

Electron. Imaging., vol.25, 013013, 2016. doi:10.1117/1.JEI.25.1.013013.

[23] Y.H. Chan and S. M. Cheung, "Feature preserving multiscale error diffusion for digital halftoning," Journal of Electronic Imaging, vol.13, pp.639-645, 2004.

[24] Y.H. Fung and Y.H. Chan, "Optimizing the error diffusion filter for blue noise halftoning with multiscale error diffusion," IEEE Trans. Image Process., vol.22, pp.413-417, 2013.

[25] R. Ulichney, Digital Halftoning, MIT Press, Cambridge, MA, 1987

[26] P. Scheunders and S. DeBacker, "Joint quantization and error diffusion of images using competitive learning," Proc. of ICIP, pp.811–814, 1997.

[27] D. Özdemir and L. Akarun, "Fuzzy error diffusion," IEEE Trans. Image Process., vol.9, pp.683–690, 2000.

[28] D. Özdemir and L. Akarun, "Fuzzy algorithms for combined quantization and dithering," IEEE Trans. Image Process., vol.10, pp.923-931, 2001.

[29] N. Nikolaidis and I. Pitas, "Robust image watermarking in the spatial domain," Signal Processing, vol.66, no.3, pp.385–403, 1998.

[30] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Proc. of Computer Vision, pp.839– 846, 1998.

[31] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., pp.679–698, 1986.

[32] J. M. Guo and Y. F. Liu, "Joint compression/ watermarking scheme using majority-parity guidance and halftoning-based block truncation coding", IEEE Trans. Image Process., vol.19, no.8, pp.2056-2069, 2010.

[33] Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Trans. Image Process., vol.13, pp.600–612, 2004.

[34] J. B. Mulligan and A. J. Ahumada Jr., 'Principled halftoning based on human vision models', Proc. of SPIE Human Vision, vol.1666, pp.109-121, 1992.

[35] X. Zhang and B.A. Wandell, "A spatial extension of CIELAB for digital color image reproduction", Journal of the Society for Information Display, vol.5, pp.61-63, 1997.

[36] M. Hassan and C. Bhagvati, "Structural similarity measure for color images," International Journal of Computer Applications, vol.43, pp.7-12, 2012.

[37] H. Chang, H. Yang, Y. Gan and M. Wang, "sparse feature fidelity for perceptual image quality assessment", IEEE Trans. Image Process., vol.22, pp.4007-4018, 2013.

[38] X. Zhang, D. A. Silverstein, J. E. Farrell, and B. A. Wandell, "Color image quality metric S-CIELAB and its application on halftone texture visibility," Proc. of IEEE COMPCON, pp.44–48, 1997.

Figure caption list

Figure 1. Block diagrams showing the flow of (a) color-to-grayscale conversion, and (b) grayscale-to-color mapping

Figure 2. Different domains for palette generation: (a) RGB color cube in *rgb* space, (b) transformed RGB color cube in *xyz* space and (c) the *x*-*y* plane used to derive a palette color

Figure 3. An example showing how 6 consecutive palette colors in our palette form an enclosure to cover most image pixel colors having similar luminance levels. The palette is developed based on the image shown in Figure 4(a).

Figure 4. Results of using different palettes in the proposed RCGC algorithm to reconstruct a color image: (a) original, (b) a palette obtained with fuzzy clustering [18] and (c) the proposed palette.

Figure 5. Enclosures formed with d palette colors when the proposed palette is developed under different settings: (a) d=3, (b) d=6, and (c) d=12. The data is obtained for k=100 and the palettes are developed for the image shown in Figure 4(a).

Figure 6. Recovering color image from our color-embedded grayscale image: (a) color-embedded grayscale image $I_{\hat{C}}$; (b) edge map obtained with Fig. 6(a); (c) edge map obtained with the original image C; (d) color image obtained before non-linear low-pass filtering, $\hat{C}$; (e) color image obtained after nonlinear low-pass filtering, $\tilde{C}$; (f) original color image C.

Figure 7. Block diagram of color image reconstruction with halftone artifacts suppression

Figure 8. The plot of J vs. σ and μ

Figure 9. Test image set 2

Figure 10. Color-embedded grayscale images obtained with various algorithms (Test image: Woman)

Figure 11. Recovered color images obtained with various algorithms (Test image: Woman)

Figure 12. Ground truth for Figures 10 and 11 (Test image: Woman)

Figure 13. Palettes associated with (a) Figure 11(d), (b) Figure 11(e) and (c) Figure 11(f). Their sizes are 256, 256 and 247 respectively.

Figure 14 Distributions of image pixel colors and our palette colors in *x*-*y* space for different testing images in Kodak image set. Green dots and blue dots denote the image pixel colors and the palette colors respectively.

Figure 15. Color-embedded grayscale images obtained with various algorithms (Test image: Caps)

Figure 16. Recovered color images obtained with various algorithms (Test image: Caps)

Table caption list

Table 1. Performance of RCGC algorithms for Kodak image set [13]

Table 2. Performance of RCGC algorithms for test image set 2

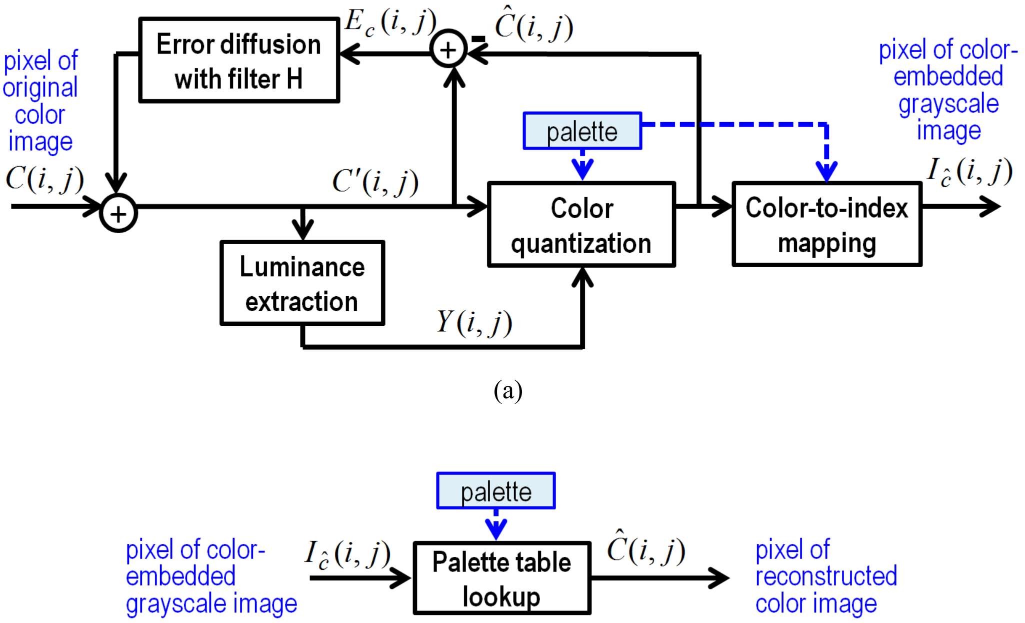
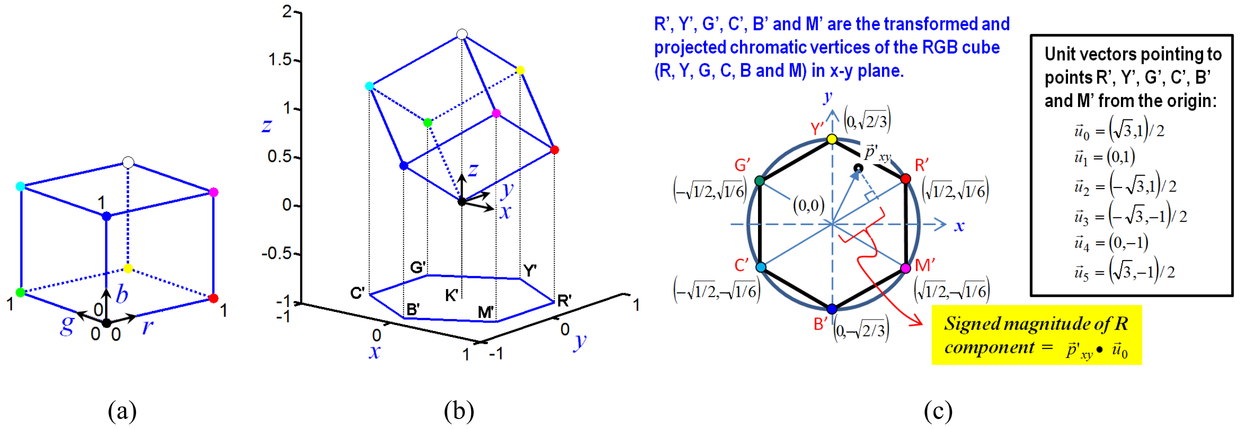


Figure 1 Block diagrams showing the flow of (a) color-to-grayscale conversion, and (b) grayscale-to-color mapping

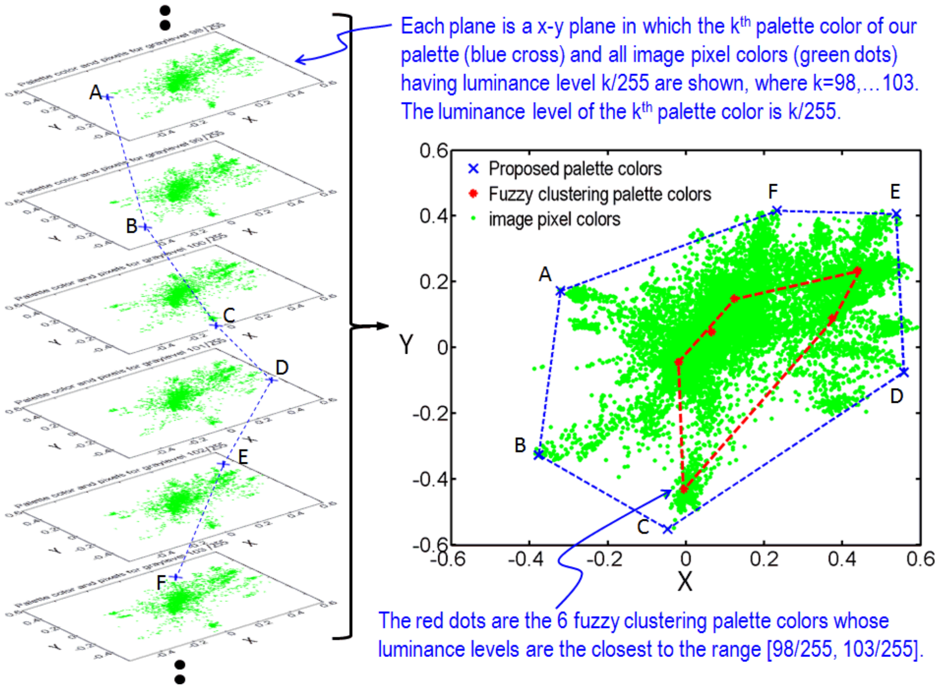

1
1
1
0
0
0
b
g
r
2
1.5
1
0.5
0
-0.5
-1
z
G'
Y'
C'
K'
B'
M'
R'
-1
0
1
x
-1
0
1
y
z
y
x
R', Y', G', C', B' and M' are the transformed and projected chromatic vertices of the RGB cube (R, Y, G, C, B and M) in x-y plane.
y
x
Y'
G'
R'
C'
M'
B'
$(0,\sqrt{2/3})$
$(-\sqrt{1/2},\sqrt{1/6})$
$(\sqrt{1/2},\sqrt{1/6})$
$(0,0)$
$(-\sqrt{1/2},-\sqrt{1/6})$
$(\sqrt{1/2},-\sqrt{1/6})$
$(0,-\sqrt{2/3})$
$\vec{p}'_{xy}$
Unit vectors pointing to points R', Y', G', C', B' and M' from the origin:
$\vec{u}_0=(\sqrt{3},1)/2$
$\vec{u}_1=(0,1)$
$\vec{u}_2=(-\sqrt{3},1)/2$
$\vec{u}_3=(-\sqrt{3},-1)/2$
$\vec{u}_4=(0,-1)$
$\vec{u}_5=(\sqrt{3},-1)/2$
Signed magnitude of R component $= \vec{p}'_{xy}\bullet\vec{u}_0$

(a)

(b)

(c)

Figure 2 Different domains for palette generation: (a) RGB color cube in *rgb* space, (b) transformed RGB color cube in *xyz* space and (c) the *x*-*y* plane used to derive a palette color


Each plane is a x-y plane in which the kth palette color of our palette (blue cross) and all image pixel colors (green dots) having luminance level k/255 are shown, where k=98,…103. The luminance level of the kth palette color is k/255.

The red dots are the 6 fuzzy clustering palette colors whose luminance levels are the closest to the range [98/255, 103/255].

Figure 3. An example showing how 6 consecutive palette colors in our palette form an enclosure to cover most image pixel colors having similar luminance levels. The palette is developed based on the image shown in Figure 4(a).

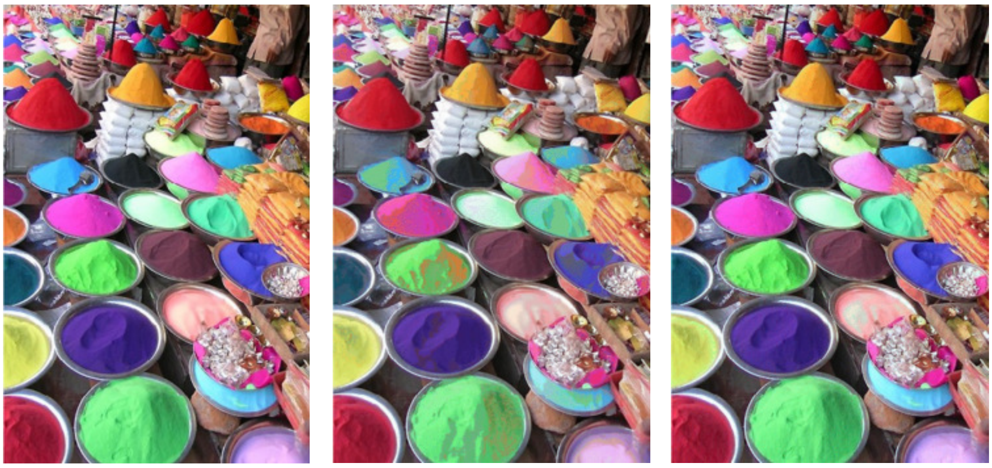
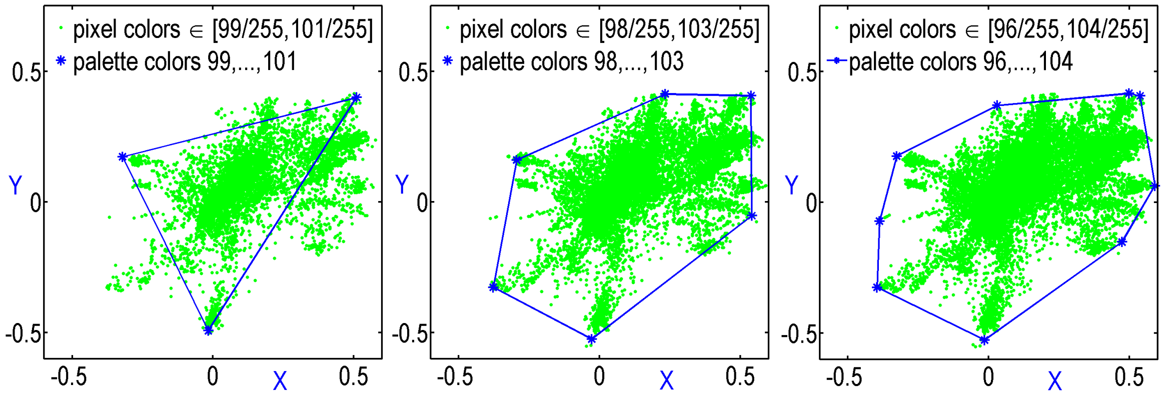
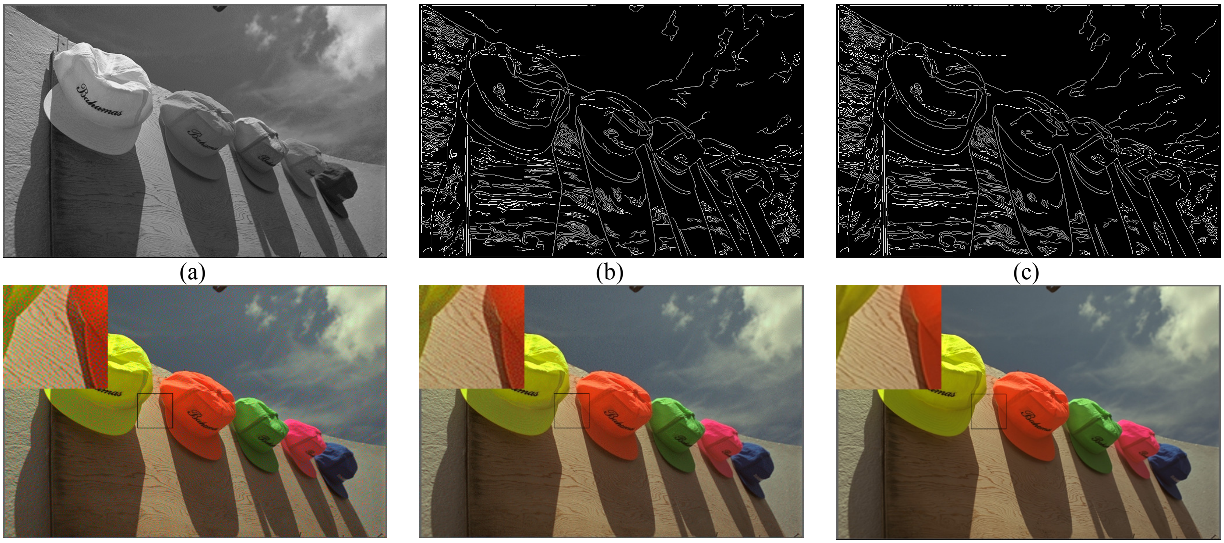

(a) (b) (c)

Figure 4. Results of using different palettes in the proposed RCGC algorithm to reconstruct a color image: (a) original, (b) a palette obtained with fuzzy clustering [18] and (c) the proposed palette.



(a) (b) (c)

Figure 5. Enclosures formed with *d* palette colors when the proposed palette is developed under different settings: (a) *d*=3, (b) *d*=6, and (c) *d*=12. The data is obtained for *k*=100 and the palettes are developed for the image shown in Figure 4(a).



(a) (b) (c)

(d) (e) (f)

Figure 6 Recovering color image from our color-embedded grayscale image: (a) color-embedded grayscale image $I_{\hat{C}}$; (b) edge map obtained with Fig. 6(a); (c) edge map obtained with the original image *C*; (d) color image obtained before non-linear low-pass filtering, $\hat{C}$; (e) color image obtained after nonlinear low-pass filtering, $\tilde{C}$; (f) original color image *C*.

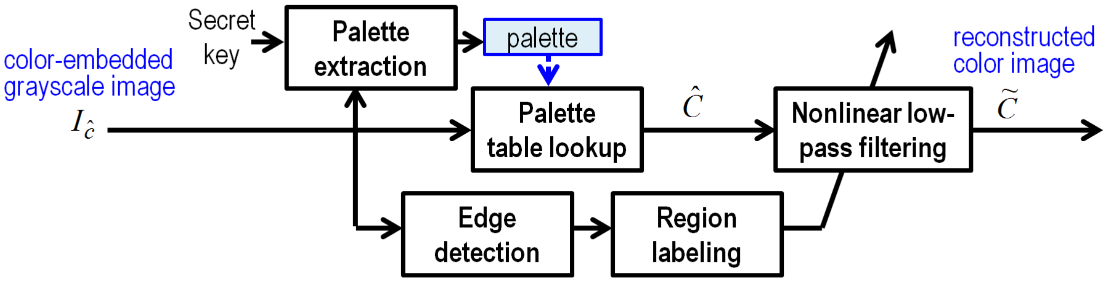


Figure 7 Block diagram of color image reconstruction with halftone artifacts suppression

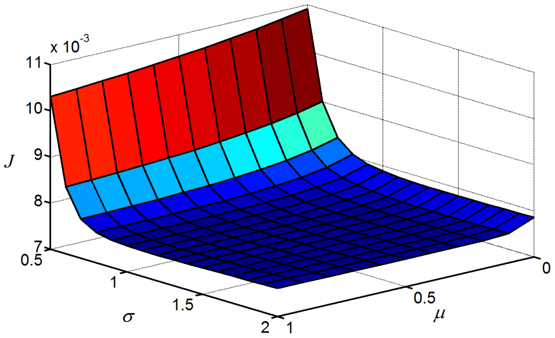


Figure 8 The plot of *J* vs. σ and μ

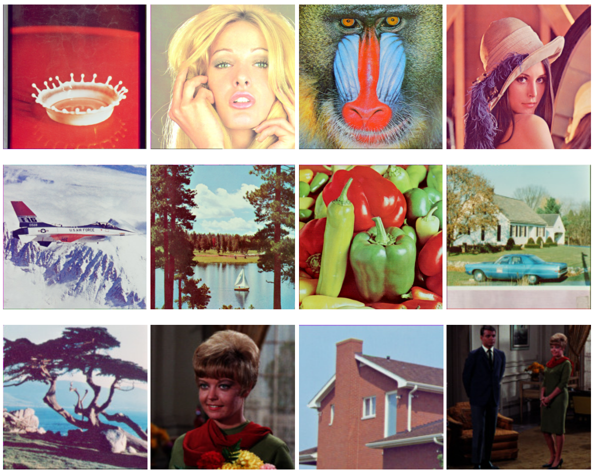
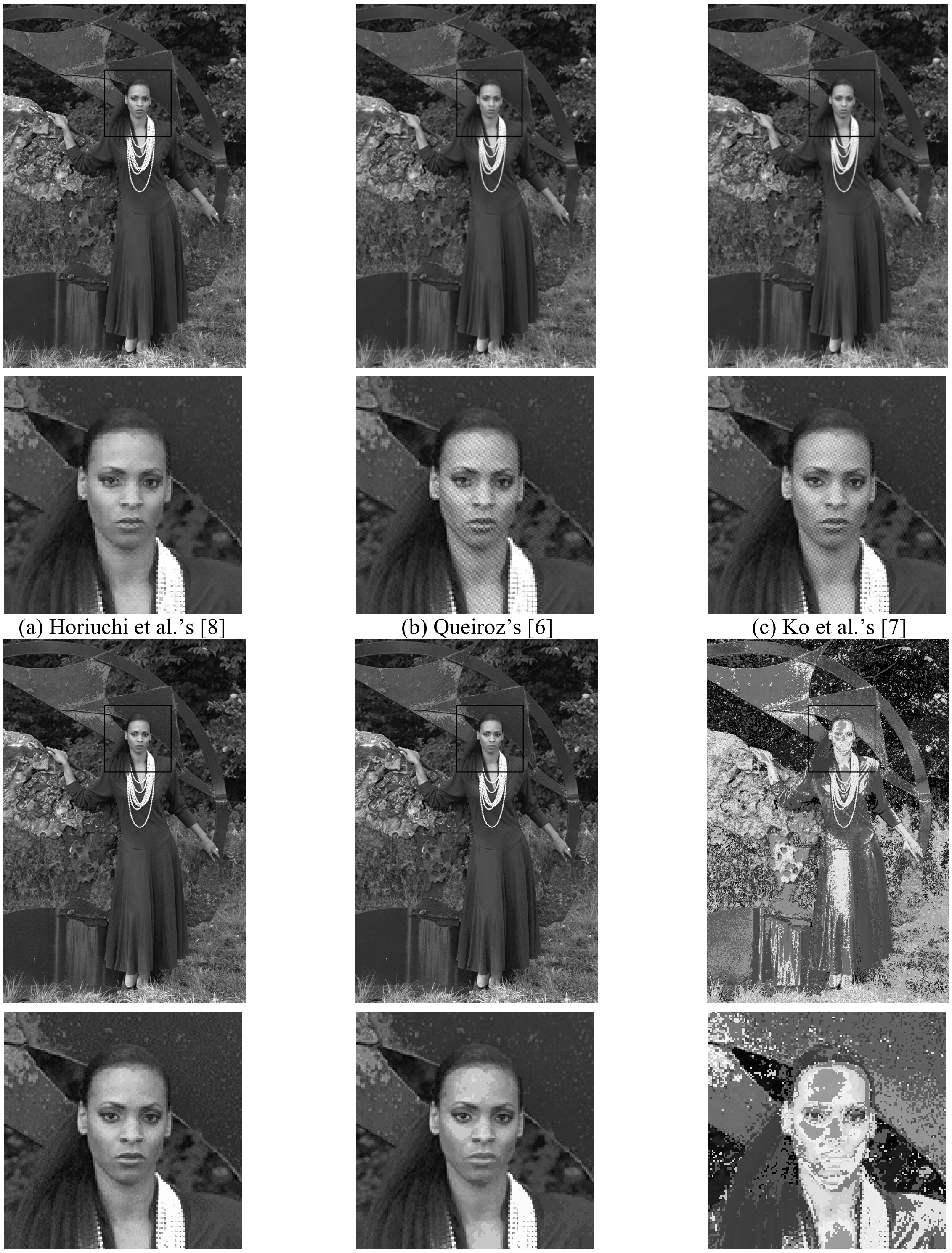


Figure 9 Test image set 2



(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

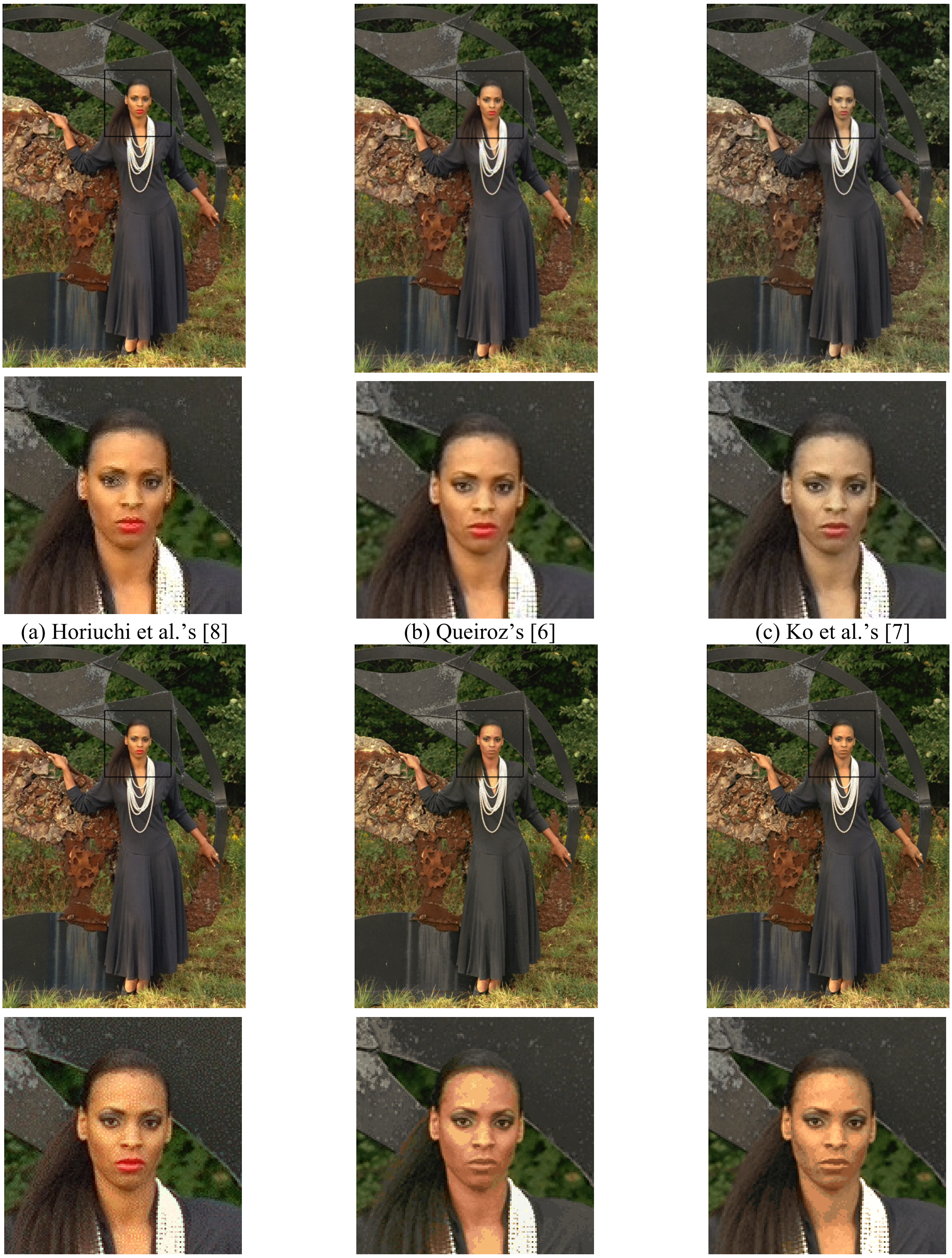
(c) Ko et al.'s [7]

(d) Ours

(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

Figure 10. Color-embedded grayscale images obtained with various algorithms (Test image: Woman)



(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

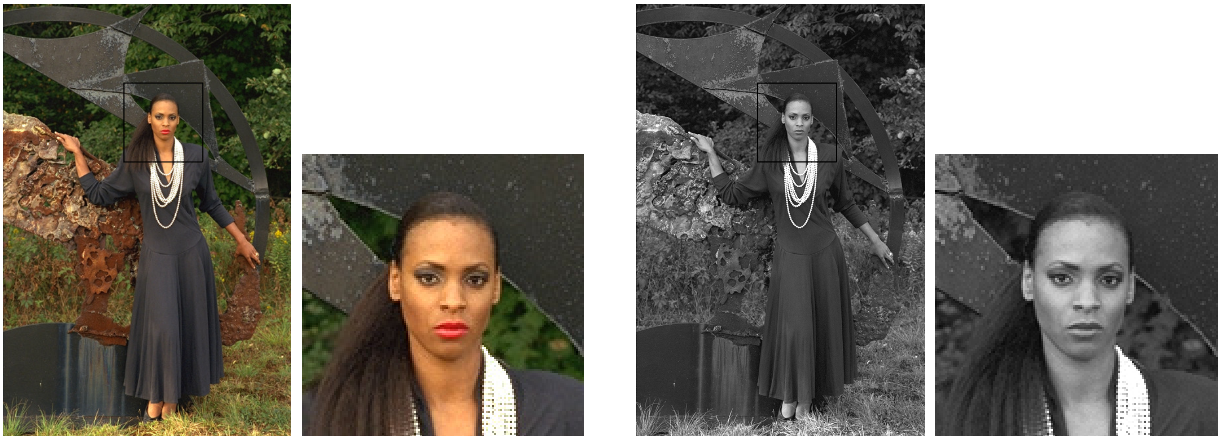
(c) Ko et al.'s [7]

(d) Ours

(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

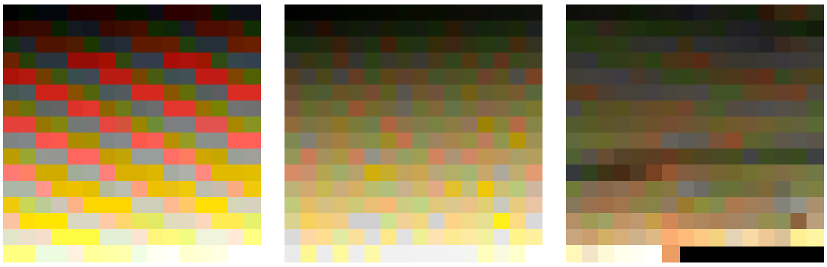
Figure 11. Recovered color images obtained with various algorithms (Test image: Woman)



(a) Original

(b) Luminance plane of original

Figure 12. Ground truth for Figures 10 and 11 (Test image: Woman)



(a)

(b)

(c)

Figure 13. Palettes associated with (a) Figure 11(d), (b) Figure 11(e) and (c) Figure 11(f). Their sizes are 256, 256 and 247 respectively.

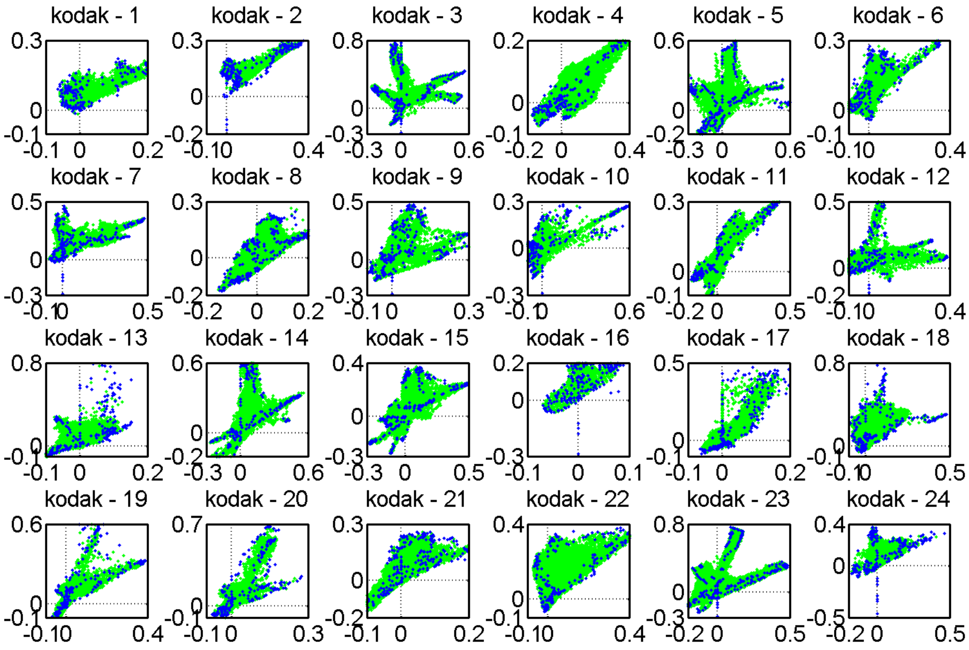
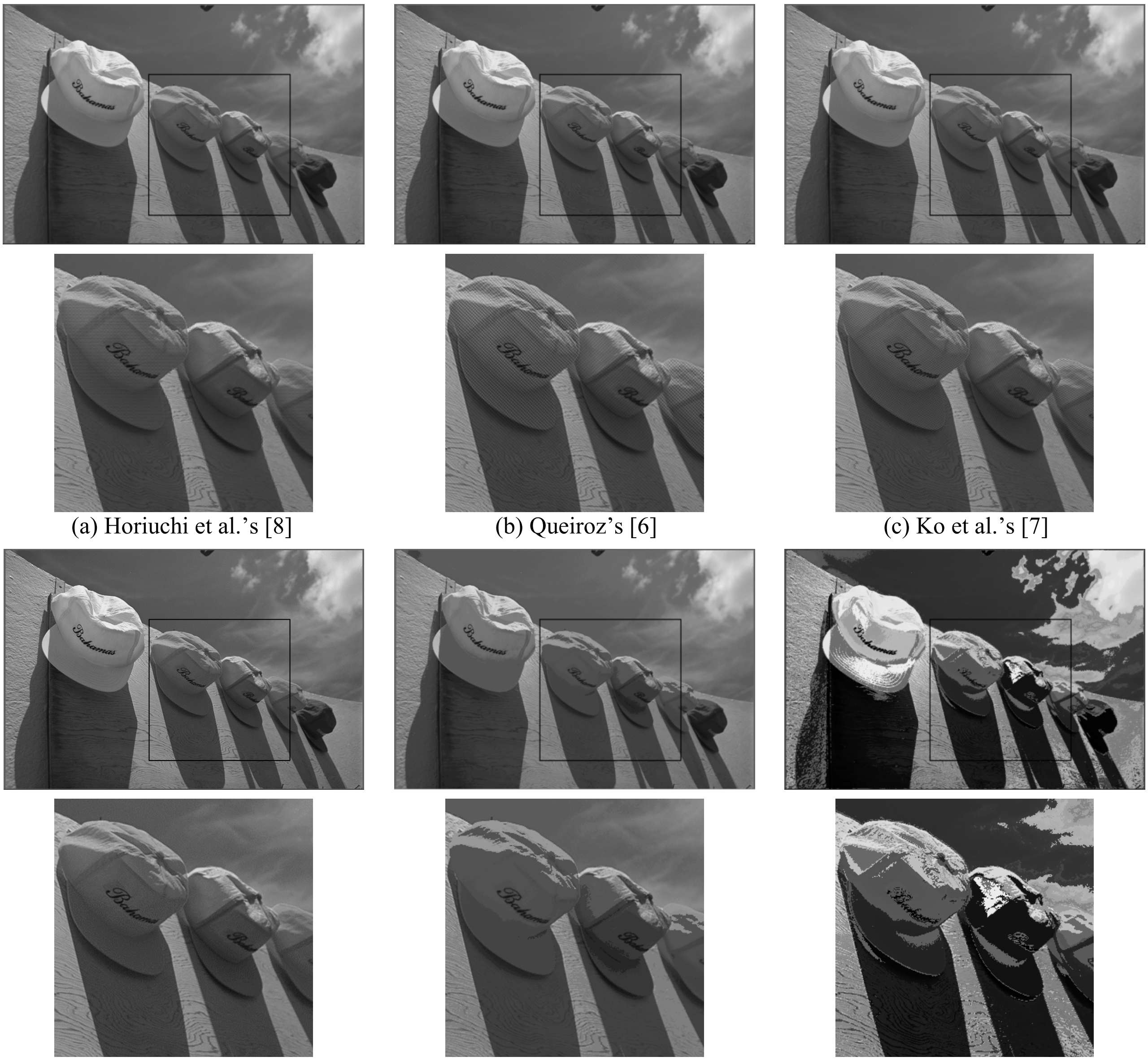


Figure 14 Distributions of image pixel colors and our palette colors in *x*-*y* space for different testing images in Kodak image set. Green dots and blue dots denote the image pixel colors and the palette colors respectively.



(a) Horiuchi et al.’s [8]

(b) Queiroz’s [6]

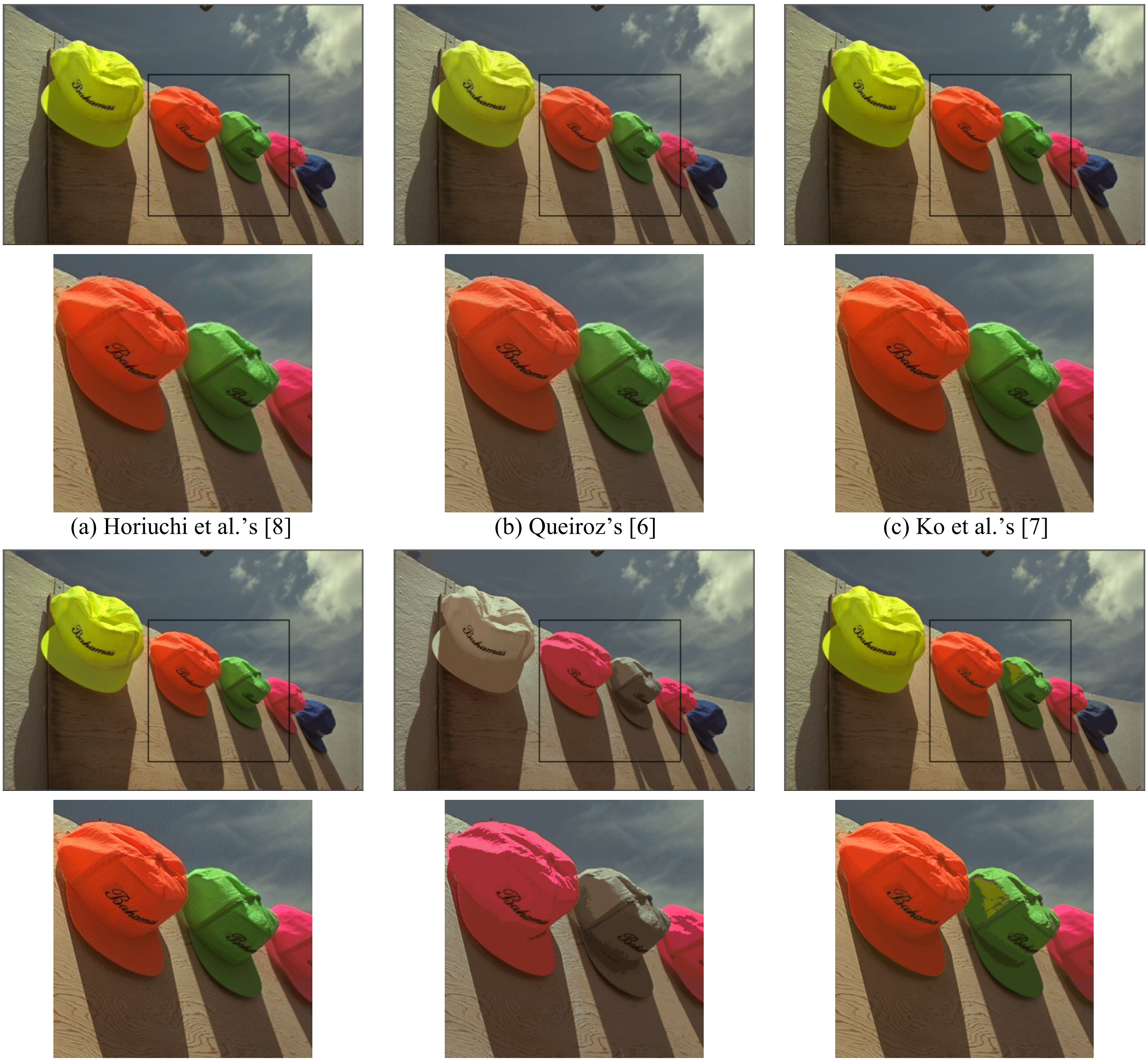
(c) Ko et al.’s [7]

(d) Ours

(e) Tanaka et al.’s [14]

(f) Chaumonta et al.’s [4]

Figure 15. Color-embedded grayscale images obtained with various algorithms (Test image: Caps)



(a) Horiuchi et al.'s [8]

(b) Queiroz's [6]

(c) Ko et al.'s [7]

(d) Ours

(e) Tanaka et al.'s [14]

(f) Chaumonta et al.'s [4]

Figure 16. Recovered color images obtained with various algorithms (Test image: Caps)

Method	*Color-embedded Grayscale Image*				*Recovered Color Image*							
	PSNR (dB)	*SSIM*	*dpi = 100*	*dpi = 150*	*CPSNR* (dB)	*SSF*	*dpi = 100*			*dpi = 150*		
			HVS-PSNR (dB)	*HVS-PSNR (dB)*			*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*	*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*
Queiroz [6]	31.64	0.9760	40.92	43.85	31.84	**0.9998**	36.78	0.877	1.754	38.09	0.889	1.630
Ko et al. [7]	32.82	0.9896	**42.88**	**46.67**	32.13	0.9995	36.05	0.838	3.433	37.01	0.848	3.272
Horiuchi et al. [8]	33.20	**0.9997**	39.19	43.66	29.53	0.9997	33.69	0.927	1.488	36.02	0.952	1.214
Tanaka et al. [14]	28.07	0.9866	29.79	31.19	33.07	0.9986	34.78	0.874	2.547	35.51	0.879	2.435
Chaumont et al. [4]	16.72	0.6432	17.92	18.47	**37.79**	**0.9998**	**39.94**	0.962	1.017	40.99	0.968	0.921
Proposed	**35.87**	0.9883	41.54	44.01	32.71	**0.9998**	38.04	**0.963**	**0.790**	**41.24**	**0.968**	**0.576**

Table 1 Performance of RCGC algorithms for Kodak image set [13]

Method	*Color-embedded Grayscale Image*				*Recovered Color Image*							
	PSNR (dB)	*SSIM*	*dpi = 100*	*dpi = 150*	*CPSNR* (dB)	*SSF*	*dpi = 100*			*dpi = 150*		
			HVS-PSNR (dB)	*HVS-PSNR (dB)*			*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*	*HVS-PSNR$_{Color}$ (dB)*	*CSSIM*	*$\Delta E_{S-CIELAB}$*
Queiroz [6]	29.89	0.9615	39.59	43.12	28.68	0.9987	30.32	0.716	3.111	31.22	0.747	2.845
Ko et al. [7]	30.26	0.9799	41.18	**46.06**	29.88	0.9981	31.42	0.823	2.509	32.18	0.856	2.192
Horiuchi et al. [8]	**35.95**	**0.9998**	40.87	43.24	29.06	0.9985	30.97	0.841	2.323	31.99	0.877	1.961
Tanaka et al. [14]	31.31	0.9743	32.75	33.56	33.24	0.9972	34.85	0.874	1.916	37.22	0.886	1.776
Chaumont et al. [4]	16.03	0.6271	16.96	17.35	**34.41**	0.9987	**36.85**	0.919	1.481	38.32	0.929	1.335
Proposed	**35.95**	0.9861	**41.33**	43.51	31.47	**0.9995**	36.45	**0.956**	**0.966**	**39.52**	**0.964**	**0.727**

Table 2 Performance of RCGC algorithms for test image set 2