Multiscale Probabilistic Dithering for Suppressing Contour Artifacts in Digital Images

Sitaram Bhagavathy, *Member, IEEE*, Joan Llach, *Member, IEEE*, and Jiefu Zhai, *Member, IEEE*

***Abstract*—A method is proposed for reducing the visibility of "contour artifacts," i.e., false contours resulting from color quantization in digital images. The method performs a multiscale analysis on the neighborhood of each pixel, determines the presence and scale of contour artifacts, and probabilistically dithers (perturbs) the color of the pixel. The overall effect is to "break down" the false contours, making them less visible. The proposed method may be used to reduce contour artifacts at the same bit depth as the input image or at higher bit depths. The contour artifact detection mechanism ensures that artifact-free regions remain unaffected during the process.**

***Index Terms*—Contour artifact removal, decontouring, false contour removal.**

I. Introduction

DEPENDING on the application, digital images are represented at various bit depths, e.g., 8 bits per pixel (bpp) for most consumer applications, 10 bpp for some professional applications, and so on. A higher bit depth allows more colors to be represented and, therefore, a higher visual quality. Often, however, the colors in digital images have to be requantized in order to reduce the bit depth, e.g., when a 10-bpp image needs to be shown on an 8-bpp display or compressed with an 8 bpp encoder. Reducing the bit depth results in a smaller number of available (discrete) colors in the palette. With the smaller number of colors, it may not be possible to represent some areas with continuous color variation such that this continuity can be perceived by the human visual system. These areas, therefore, break up into a number of "bands" of constant color, with a small color difference between adjacent bands (corresponding to the smallest color difference allowed by the final bit-depth). Boundaries between the resulting bands may be visible as false contours, leading to artifacts collectively referred to as "contour artifacts" [see Fig. 5(a)].

There are two main approaches for reducing visible false contours as a result of bit depth reduction in digital images. The first one affects the images before or during the requantization process. This may involve adding noise to the image prior to quantization [1], diffusing the quantization error among neighboring pixels [2], [3], or a feedback-based quantization strategy [4]. Error diffusion methods [2], [3] are widely used during bit depth reduction in order to mitigate the occurrence of false contours. Despite their application, however, false contours are often introduced in the process. The second approach affects the image *after* quantization. This becomes necessary in order to reduce the visibility of contour artifacts that have already appeared as a result of bit depth reduction. The proposed method belongs to the latter category of *contour artifact removal* (CAR) methods.

CAR methods take an input image at a bit depth N containing false contours and output an image at a bit depth $M(\geq N)$ wherein false contours are visibly reduced. The two major problems to be tackled by CAR methods are the detection of regions containing contour artifacts, and the reduction of these artifacts. Lee *et al.* [5] proposed a two-stage false contour detection algorithm that first eliminates smooth regions (not containing texture, edges, or false contours) and then separates false contours from edges and texture using directional contrast features. The false contours are reduced by applying 1-D directional smoothing filters whose directions are orthogonal to that determined by the directional contrast features.

Ahn *et al.* [6] detect flat regions, or regions of low frequency content, since these have a higher likelihood of containing contour artifacts. A random shuffle is then applied to each pixel in the flat regions. This process exchanges the color of the pixel with that of a random pixel in its neighborhood. The downside of the shuffle is that small details in almost flat regions (e.g., stars in a sky region) may be spread out or lost. This is a possibility since regions that are mostly flat except for small details may be wrongly detected as flat regions.

Daly *et al.* [7] propose a predictive cancellation algorithm for "decontouring." This method predicts where false contours will occur by low-pass filtering the input image (resulting in image A at bit depth $P > N$) and then quantizing the filtered image (resulting in image B at bit depth N). The difference between image A and image B (at bit depth P), containing the predicted contours, is then subtracted from the input image. This results in an output image (at bit depth $M = P$) with fewer false contours. Daly *et al.* try to remove false contours without adding noise or dither to the image. This is desirable in the case of noise-free images, such as images generated by computer graphics and line art with gradients.

Daly [8] subsequently proposed the multiband coring algorithm, wherein false contours are suppressed by removing low amplitudes in the high-pass band of the image. The image is separated into low-pass and high-pass components by applying

Manuscript received May 16, 2008; revised March 23, 2009. First published May 05, 2009; current version published August 14, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Eric Kolaczyk.

The authors are with the Signal Acquisition and Processing Group, Thomson Corporate Research, Princeton, NJ 08540 USA (e-mail: sitaram.bhagavathy@thomson.net; joan.llach@thomson.net; jiefu.zhai1@thomson.net).

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

Digital Object Identifier 10.1109/TIP.2009.2022293

a low-pass filter and subtracting the result from the original image, respectively. Thereafter, the low-amplitude signals in the high-pass band are suppressed by applying a nonlinear coring function. The coring function is made adaptive to the local spatial activity in order to avoid removing valid low-amplitude high spatial frequencies.

Since image compression is a form of quantization, most CAR methods have potential use in postprocessing for compression artifact removal. Indeed, many of the methods described above are motivated in part by their application to postprocessing images before they are displayed. Some of these are even inspired by previous work on compression artifact removal (such as [9] and [10]).

Some of the above described methods make use of smoothing (low-pass) filters for reducing false contours. Smoothing filters can be used only when the bit depth of the output is greater than that of the input (i.e., $M > N$), so that the output image can represent the intermediate colors created by the smoothing process. Therefore, the methods of Lee *et al.* [5], Daly *et al.* [7], [8] are not applicable when the output image has to be of the same bit depth as the input image (which is a requirement in some applications).

Furthermore, the prior methods do not address the issue of the scale of contour artifacts. The term "scale of contour artifacts" refers to the degree of proximity between adjacent false contours in a local region. By convention, larger scales shall stand for a greater distance between false contours. For example, the contour artifacts in Fig. 5(a) decrease in scale (i.e., become denser) from the left to the right. This issue is important since false contours which are closely bunched together will need to be handled differently from those which are far apart. In [5], [7], and [8], closer contours will necessitate smoothing filters with smaller supports, and in [6], closer contours will necessitate a smaller neighborhood for the random shuffler. However, the scale of contour artifacts is not determined in any of the above methods.

The method we propose performs a multiscale analysis on the neighborhood of each pixel, and determines the presence and scale of contour artifacts around that pixel. Thereafter, the color of each pixel in the artifact regions is probabilistically dithered based on the distribution of colors in its neighborhood (of appropriate scale). The overall effect is to "break down"[1] the false contours making them less visible. The proposed method has the following advantages:

- effective suppression of contour artifacts of different scales;
- contour artifacts can be reduced at the same bit depth as the input image or at higher bit depths;
- low amplitude signal-dependent dithering reduces the visibility of false contours with a relatively low level of noise addition; and
- preservation of fine detail in the image, including small details occurring in almost flat regions.

The paper is organized as follows. Section II describes the method proposed for multiscale contour artifact detection. Section III describes the proposed probabilistic dithering strategy and methods based on this for reducing the visibility of the detected artifacts. Section IV describes the process of bit depth extension, wherein the bit depth of the output image is greater than that of the input image. Section V demonstrates the effectiveness and advantages of the proposed approach by presenting experimental results. Section VI concludes with a discussion of the current work and future directions.

II. Multiscale Contour Artifact Detection

With regard to reducing contour artifacts, it is in general desirable to alter only regions where such artifacts occur. Furthermore, contour artifacts may occur at various scales and estimating the scale helps in effectively reducing such artifacts. Therefore, the first step in contour artifact reduction is to detect the regions wherein false contours are likely to be present and the scale of the artifact therein. We propose a method for detecting the presence and scale of contour artifacts in the neighborhood of each pixel. The method involves a multiscale analysis of the color distribution in the neighborhood of a pixel.

The input to the method is an image I. The symbol, $I(x,y)$, denotes the value or color of the pixel with coordinates (x,y). At each pixel location, we first estimate the most likely scale of contour artifacts around it and then decide whether or not significant artifacts are present at that scale. This process is as follows. Let (x,y) be the pixel currently being analyzed. Let us consider a number of scales, $s = 1, 2, \ldots, S$, each of which corresponds to a neighborhood, $N_s(x,y)$, around the pixel (x,y). As the scale index s increases, so does the size of the neighborhood N_s.

At each scale s, a confidence score, $c(s)$, for the likelihood of contour artifacts is computed as follows:

$$c(s) = p(0,s) \times \mathrm{MAX}\left[\frac{p(-1,s)}{p(0,s)+p(-1,s)}, \frac{p(1,s)}{p(0,s)+p(1,s)}\right] \tag{1}$$

where $p(k,s)$ refers to the fraction (or probability) of all pixels in $N_s(x,y)$ having the value $I(x,y) + k$. In other words, $p(\pm 1, s)$ refers to the fraction of all pixels in the neighborhood differing by ± 1 from the center pixel value $I(x,y)$. Similarly, $p(0,s)$ refers to the fraction of all pixels in the neighborhood having the same value as that of the center pixel. The term $\mathrm{MAX}[a,b]$ refers to the greater of a and b. The second term in the RHS of (1) gives a measure of the likelihood of contour artifacts being present. Since a false contour is the result of quantizing a smooth gradient, pixels on either side of the contour differ[2] by a value of 1. If $N_s(x,y)$ overlaps one or more contours, the second term in the RHS of (1) is likely to have a high value due to a relatively high number of pixels differing by 1 from the center pixel value. The first term in the RHS of (1) measures how significant the contour effect is. If the fraction of affected pixels, $p(0,s)$, is low, then it is likely to be visually insignificant.

[1]This phrase was used in [1].

[2]Note that, in this paper, we assume the false contours to be caused by simple color quantization. If color quantization is preceded or followed by other operations (e.g., in image compression), the pixels on either side of a contour may differ by a value greater than 1. This possibility may be handled (at the expense of losing some fine detail) by using $p(\pm b, s)$ (where $b > 1$) instead of $p(\pm 1, s)$ in (1).

It is desirable to avoid affecting regions with low likelihood of contour artifacts. To this end, before we choose the "best" (most representative) scale of the artifact at (x, y), we eliminate scales which have a low likelihood of representing contour artifacts. This is done by applying some criteria on the probabilities $p(k, s)$ where $k \in \{-1, 0, 1\}$. In our implementation, we *detect* the presence of contour artifacts at pixel (x, y) at scale s if

$$p(0, s) > T \text{ and } [p(-1, s) > T \text{ or } p(1, s) > T] \qquad (2)$$

where $T \in (0, 0.5)$ is a preset threshold (typically, $T \geq 0.1$). If none of the scales, s, obey the criteria in (2), we assume that the pixel (x, y) is not part of a contour artifact. If artifacts are detected at one or more scales at (x, y), we choose the scale, $s^*(x, y)$, with the highest confidence score among all scales at which artifacts are detected, i.e., $c^* = c(s^*)$, as the *representative scale* of the artifact at (x, y). The representative scales, $s^*(x, y)$, of pixels in the artifact regions are then used by the probabilistic dithering algorithm (Section III) to reduce the visibility of the artifacts. Henceforth, we say that a contour artifact at pixel (x, y) has scale $s^*(x, y)$, if the latter is the representative scale at the pixel.

Fig. 1 illustrates how the above detection scheme works for different types of image neighborhoods (corresponding to some scale, say s). The centers of the neighborhoods are marked with an "x." In the case of the *flat* region shown in Fig. 1(a), $p(0, s) = P(c) = 1$, $p(-1, s) = p(1, s) = 0$, and, therefore, no contour artifact is detected according to (2). For the *step* in Fig. 1(b), $p(0, s) = P(c) \approx 0.5$ and $p(1, s) = P(c + 1) \approx 0.5$, and, therefore, a contour artifact is detected [for typical values of T in (2)]. In rare cases, such a step is not an artifact but an intended image pattern (see Section VI). If the neighborhood contains a gradual *slope* [shown in Fig. 1(c)] or a typical *texture*, no artifact is detected as the neighborhood is shared among a large number of color values (i.e., $p(0, s)$, $p(1, s)$, and $p(-1, s)$ have low values). Note that the range of neighborhood sizes (i.e., the set of scales, S) to consider is critical for discriminating between contour artifacts and other local image phenomena.

In some situations, however, the detection framework described above may fail. Some particular image textures made of two colors differing by one level could lead to misdetection. For the example shown in Fig. 1(d), $p(0, s) = P(c) \approx 0.5$ and $p(1, s) = P(c + 1) \approx 0.5$, resulting in a contour artifact being falsely detected. This possibility arises because (1) and (2) take into account only the *statistical* distribution of pixel values and not their *spatial* distribution. The result of such false detection is that when contour artifact reduction is applied, the falsely detected textured areas are also modified and this is undesirable in most applications. The following section describes a way of reducing such false detection by introducing spatial constraints into the artifact detection process.

A. Contour Artifact Detection With Spatial Constraints

In order to reduce false detection in textured areas, spatial constraints need to be introduced into (1) and (2). Fig. 1(b) illustrates one important characteristic of contour artifacts, i.e., they occur in boundaries between flat regions in images. The pixels that form the artifact lie in regions of constant color. The magnitude of the gradient[3] at these pixels must, therefore, be very low (in theory, zero). In the light of this observation, a spatial constraint is formulated in terms of a threshold, τ_g, on the gradient magnitude at each pixel. Only pixels wherein the gradient magnitude is less than τ_g are taken into account when computing the probabilities $p(k, s)$ in (1) and (2). These probabilities are then computed as follows:[4]

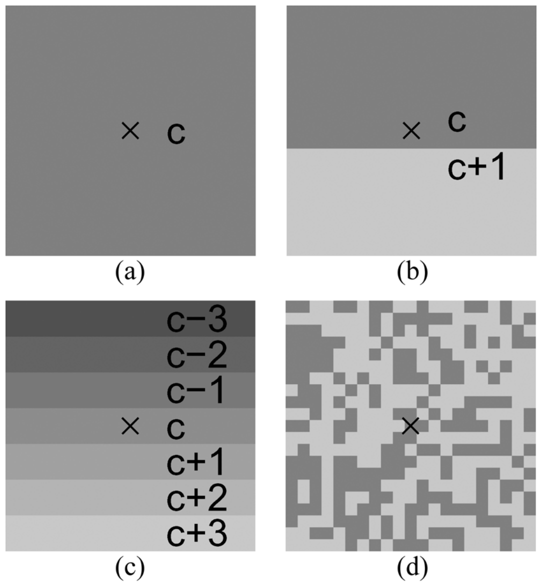

Fig. 1. Illustration of how contour artifact detection works for different types of image neighborhoods (centers marked by "×"): (a) a flat region of color c; (b) a step from color c to $c + 1$; (c) a gradual slope from color $c - 3$ to $c + 3$; and (d) a texture formed by two colors c and $c + 1$.

$$p(k, s) = \frac{\sum_{\{(x',y') \in N_s(x,y) \mid \|\nabla(x',y')\| < \tau_g\}} \delta\left(I(x', y'), I(x, y) + k\right)}{\sum_{\{(x',y') \in N_s(x,y) \mid \|\nabla(x',y')\| < \tau_g\}} 1} \qquad (3)$$

where $\|\nabla(x', y')\|$ is the gradient magnitude at pixel (x', y') and $\delta(.,.)$ is an indicator function defined as follows:

$$\delta(a, b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{otherwise.} \end{cases} \qquad (4)$$

The numerator in (3) is the number of pixels in $N_s(x, y)$ having the value $I(x, y) + k$ and satisfying the condition that their gradient magnitude is less than τ_g. The denominator is the total number of pixels in $N_s(x, y)$ that satisfy the gradient condition. The threshold, τ_g, is set to a low value between 0 and 1. The gradient constraint in (3) effectively prevents textured regions from being falsely detected as containing contour artifacts.

III. Contour Artifact Reduction by Probabilistic Dithering

At each pixel where contour artifacts have been detected, probabilistic dithering is applied in order to break down the

[3]The gradient at a pixel (x, y) is given by $\nabla(x, y) = ((\partial I(x, y))/\partial x, (\partial I(x, y))/\partial y)$. The magnitude of the gradient is given by $\|\nabla(x, y)\| = \sqrt{((\partial I(x, y))/\partial x)^2 + ((\partial I(x, y))/\partial y)^2}$.

[4]Standard set notations apply here: $\{p|q\}$ is read as *set of all p for which condition q is satisfied.*

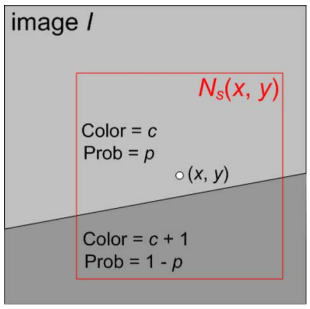

Fig. 2. Illustration of the principle of the probabilistic dithering method used for contour artifact reduction.

false contours and reduce their visibility. The idea behind probabilistic dithering is to dither or perturb the color of a pixel based on the distribution of color values in its local neighborhood. This idea is illustrated in Fig. 2.

Let $N_s(x, y)$ be a neighborhood around pixel (x, y) in image I. The neighborhood lies on a false contour formed at the boundary of two regions with color values c and $c + 1$. Let p and $1 - p$ denote the probabilities of pixels in $N_s(x, y)$ having values c and $c + 1$ respectively. In order to reduce the visibility of the false contour between the regions, the color of pixels should gradually (instead of abruptly as shown) change from c to $c + 1$ as we move from the upper region toward the lower region. However, due to the use of quantized color values, we do not have any values between c and $c+1$ to represent intermediate colors. To handle this problem, we utilize the property of the human visual system (HVS) to average the color of nearby pixels in a picture. Thus, it is possible to *perceive* a color between c and $c + 1$ at a pixel by having a certain distribution of the two colors in its immediate neighborhood. One way of achieving this effect in Fig. 2 is to assign the color value c to pixel (x, y) with probability p and $c + 1$ with probability $1 - p$. When the majority of the neighborhood lies in the upper region, the color will be c with a high probability. As it moves downward, more and more pixels will be assigned the color $c + 1$ increasing the average color gradually from c to $c + 1$.

Fig. 3 shows the result of applying probabilistic dithering to an image region containing a false contour. Fig. 3(a) is a 46 × 46-pixel region from the grayscale image in Fig. 5(e). Fig. 3(b) shows a binary representation of the region which contains two intensity values, say c and $c+1$. Fig. 3(c) shows the region after applying probabilistic dithering to the whole image (using the method that will be described in Section III-B). Fig. 3(d) shows a binary representation of the dithered region which contains two intensity values c and $c + 1$. Starting at the top-left corner, if we proceed from one side of the false contour to the other, lighter pixels $(c + 1)$ become more frequent and darker pixels (c) become less frequent. This causes a smooth variation of the local average of intensities from c at the top-left to $c + 1$ at the bottom-right. Since the HVS performs a similar averaging, we see a smooth color variation (from a distance) instead of a step after applying probabilistic dithering.

Daly *et al.* [7] point out that *white* noise is ineffective in masking false contours at admissible noise levels. This is a valid

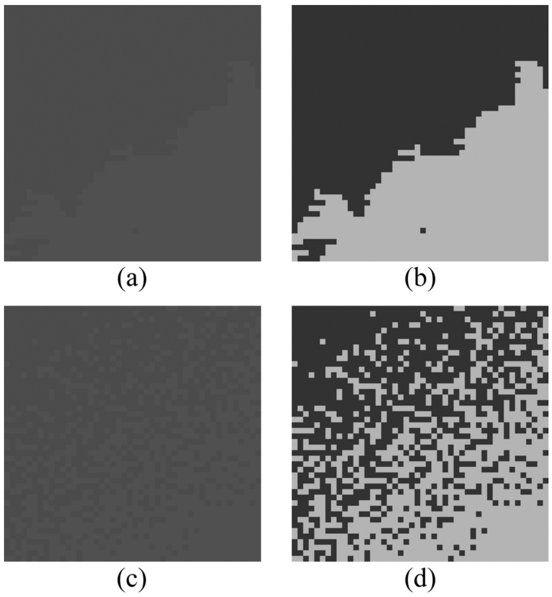

Fig. 3. Effect of probabilistic dithering on an image region: (a) a small region (46 × 46 pixels) containing a false contour from the grayscale image in Fig. 5(e); (b) binary representation of the region (the two regions differ by one intensity value in the original); (c) the region after applying probabilistic dithering to the whole image (using the method described in Section III-B); and (d) binary representation of the dithered region. (Note that these images are best viewed in the electronic pdf version.)

observation since from a distance, white noise is averaged out by the human visual system resulting in the reappearance of the false contours. However, the proposed method is quite effective in masking false contours since it applies a low-amplitude *signal-dependent* dithering in contrast with white noise that has a signal-independent distribution.

Based on the basic probabilistic dithering strategy described above, several methods may be devised for reduction of contour artifacts. In the following, we shall describe two proposed methods that provide a good level of artifact reduction. The first method is a simple extension of the basic ideas described above and the second method involves the computation of the expected mean color value at each pixel in the artifact region in order to simulate that color.

A. Probabilistic Dithering Method 1

Let J be the output of the contour artifact reduction process, i.e., the "decontoured" image. Let (x, y) be a pixel location where a contour artifact of scale $s^*(x, y)$ has been detected. The output value, $J(x, y)$, is obtained by dithering the input pixel value, $I(x, y)$, based on the probabilities $p(0, s^*)$, $p(-1, s^*)$, and $p(1, s^*)$. These probabilities are computed[5] as described in Section II. This method is a simple extension of the basic idea that uses only two probabilities described earlier in Section III. However, the result of this method is found to be perceptually superior to that of the basic method.

Initially, we shall normalize the probabilities $p(k, s^*)$ so that they sum to one, as follows:

[5]The neighborhood $N_{s^*}(x, y)$ used for computing the probabilities is determined by the artifact scale $s^*(x, y)$ at the pixel.

$$p'(k,s^*) = \frac{p(k,s^*)}{p(-1,s^*)+p(0,s^*)+p(1,s^*)},\ k \in \{-1,0,1\}. \tag{5}$$

Thereafter, we assign a color value to $J(x,y)$ with the following rule. $J(x,y)$ takes on the same value as $I(x,y)$ with the probability $p'(0,s^*)$, a value of $I(x,y)-1$ with the probability $p'(-1,s^*)$, and a value of $I(x,y)+1$ with the probability $p'(1,s^*)$. In practice, a random number, $r \in [0,1]$, may be generated and the following rule may be used:

$$J(x,y)=\begin{cases} I(x,y), & r < p'(0,s^*) \\ I(x,y)+1, & p'(0,s^*) \le r < p'(0,s^*)+p'(1,s^*) \\ I(x,y)-1, & r \ge p'(0,s^*)+p'(1,s^*). \end{cases} \tag{6}$$

The above procedure is applied to all pixel locations where contour artifacts have been detected to obtain the "decontoured" image, J.

B. Probabilistic Dithering Method 2

In this method, we first compute the expected mean color value at each pixel in the artifact region and then dither the pixel based on this mean value. The overall result is the perceptual simulation of the expected mean colors at each pixel in the artifact region. Again, let J be the output of the contour artifact reduction process and (x,y) be a pixel location where an artifact of scale $s^*(x,y)$ has been detected. We compute the expected mean value in the local neighborhood, $N_{s^*}(x,y)$, as follows:

$$m = p'(-1,s^*)(z-1)+p'(0,s^*)z+p'(1,s^*)(z+1) \tag{7}$$

where $z=I(x,y)$ and $p'(k,s^*)$ are the normalized probabilities computed in (5). Ideally, the output value, $J(x,y)$, should be equal to m. However, since a pixel can only take discrete values, a dithering strategy is devised as follows in order to make the neighborhood mean value approach m.

Let us define a *lower value* as $\lfloor m \rfloor$ and an *upper value* as $\lfloor m \rfloor + 1$, where $\lfloor m \rfloor$ refers to the largest integer not greater than m. Let us then define a probability $q = m - \lfloor m \rfloor$. In order to obtain a neighborhood mean close to m, the output value, $J(x,y)$, is assigned the upper value with probability q and the lower value with probability $(1-q)$. In practice, a uniform random number, $r \in [0,1]$, may be generated and the following rule may be used:

$$J(x,y) = \begin{cases} \lfloor m \rfloor + 1, & \text{if } r < q \\ \lfloor m \rfloor, & \text{if } r \ge q. \end{cases} \tag{8}$$

Note that the upper value is clipped at the maximum allowed value of $2^{b_I}-1$ (where b_I is the bit depth of I) and the lower value is clipped at zero. The above procedure is applied to all pixel locations where contour artifacts have been detected to obtain the "decontoured" image, J.

In the above described contour artifact detection and reduction methods, the input and output may be grayscale images or one component of color images. In general, color images are represented by multiple components, e.g., YUV, RGB. Bit depth reduction of such images involves separately quantizing each component of the image. Conversely, in order to reduce contour artifacts, we may apply the proposed method separately to each component of the quantized image.

IV. Bit Depth Extension

Bit depth extension is the reverse process of bit depth reduction, wherein the bit depth of an image is increased, e.g., from 8 to 10 bpp. It is a useful process in many applications, e.g., when displaying 8 bpp pictures on a 10-bit display. The additional bits may be used effectively to further suppress the visibility of existing contour artifacts. The described method in Section III-B may be modified to further reduce the visibility of contour artifacts during bit depth extension. The contour artifact detection mechanism remains the same and is performed at the input bit depth. Let the bit depth of the input image, I, be b_I, and that of the output image, J, be $b_J(> b_I)$. The smallest possible nonzero color difference of 1 between pixels in the input now translates to a difference of $d_{\min} = 2^{b_J - b_I}$ in the output.

The probabilistic dithering method of Section III is modified thus. Consider the pixel at (x,y). If no contour artifact is detected at (x,y), we set the output value $J(x,y)=d_{\min}I(x,y)$. If an artifact of scale $s^*(x,y)$ is detected at (x,y), the expected mean value m in the local neighborhood, $N_{s^*}(x,y)$, is computed by modifying (7) as follows:

$$m = d_{\min}\left[p'(-1,s^*)(z-1)+p'(0,s^*)z+p'(1,s^*)(z+1)\right]. \tag{9}$$

Based on the above value of m, the upper and lower values, and the probability q are computed in the same way as described in the previous section. The dithering step is also identical to that described in (8). Since $b_J > b_I$, the upper and lower values are closer in color at bit depth b_J than at bit depth b_I. Thus, at higher bit depths, the dithered result appears less noisy while still effectively reducing the visibility of contour artifacts.

V. Experimental Results

In this section, we demonstrate the effectiveness of the proposed contour artifact reduction methods using first a simple synthetic example and then a few real-world examples. We shall also illustrate the advantages of the proposed approach in comparison with the predictive cancellation "decontouring" method proposed in Daly *et al.* [7] (briefly described in Section I), which we consider to be a prominent method in the recent literature.

Let us first consider the synthetic image shown in Fig. 4(a). It is a 6 bits-per-pixel (bpp) grayscale image of size 256 $\times$ 256 pixels containing a set of intensity "bands" of varying width or scale. Each band differs from its adjacent bands by one intensity level, thus simulating contour artifacts arising from a quantization process. The two white dots (4 $\times$ 4 pixels each) represent small details which should ideally be preserved during the contour artifact reduction process. Note that most images nowadays have a bit depth of 8 bpp or more. However, most commonly available displays and printers can only represent up to 8 bpp. Therefore, we choose an input image at 6 bpp so that the bit depth extended result at 8 bpp [Figs. 4(e) and 5(d)] can still be viewed using a standard monitor or printer.

Fig. 4(b) shows the 8 bpp result after applying the method of Daly *et al.* Note that this method always outputs an image at a higher bit depth than that of the input image. The quality of

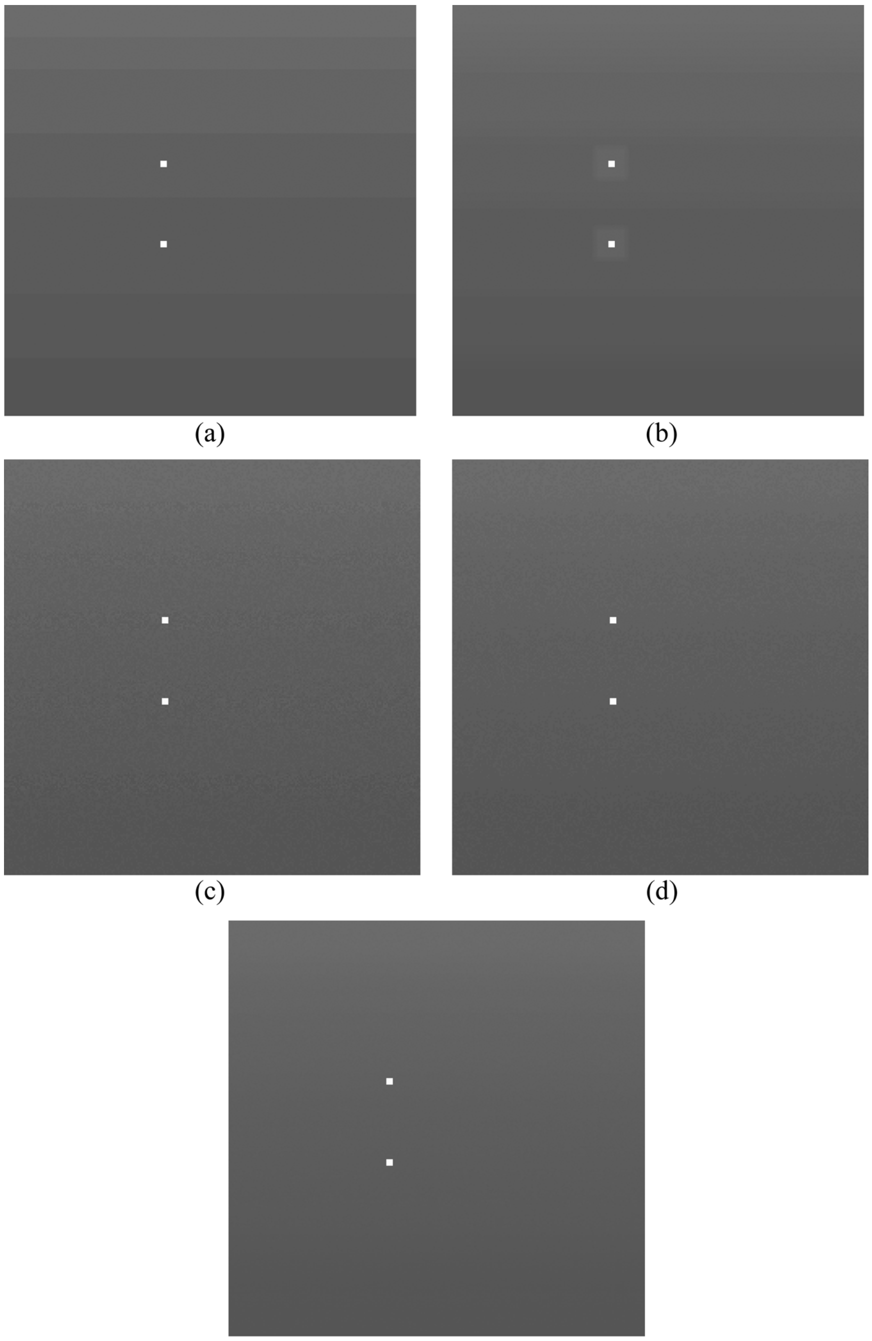

Fig. 4. (a) Input 6 bits-per-pixel (bpp) image with contours and small details (white dots); (b) result of the "decontouring" method of Daly *et al.* [7] at 8 bpp; (c) result of the method proposed in Section III-A at 6 bpp; (d) result of the method proposed in Section III-B at 6 bpp; and (e) result of the proposed bit depth extension method (Section IV) at 8 bpp. (Note that these images are best viewed in the electronic pdf version.)

the result depends on the scale of the low pass filter used. We experimented with filters of different scales and chose a 20×20 averaging filter, since it seems to be the most effective scale for this example. From Fig. 4(b), it can be observed that (i) although the method is effective in reducing existing false contours, it could introduce new ones, and (ii) the low pass filter could blur small details (the white dots).

Fig. 4(c)–(e) shows the results of applying the methods proposed in this paper. Fig. 4(c) and (d) shows the results of the methods in Section III-A and B, respectively. The dithering is performed at the same bit depth as the input image, i.e., 6 bpp. In our approach, contour artifacts can be reduced at the same bit depth as the input image or at higher bit depths. The former cannot be done using the method of Daly *et al.* From Fig. 4(c) and (d), it can be seen that the proposed methods are effective in breaking down false contours and making the artifacts less visible. The multiscale[6] contour artifact detection mechanism is effective in tackling false contours of varying scale (or local den-

[6]Note that we use 6 scales in all our experiments. The neighborhood sizes range from 10×10 pixels to 110×110 pixels.

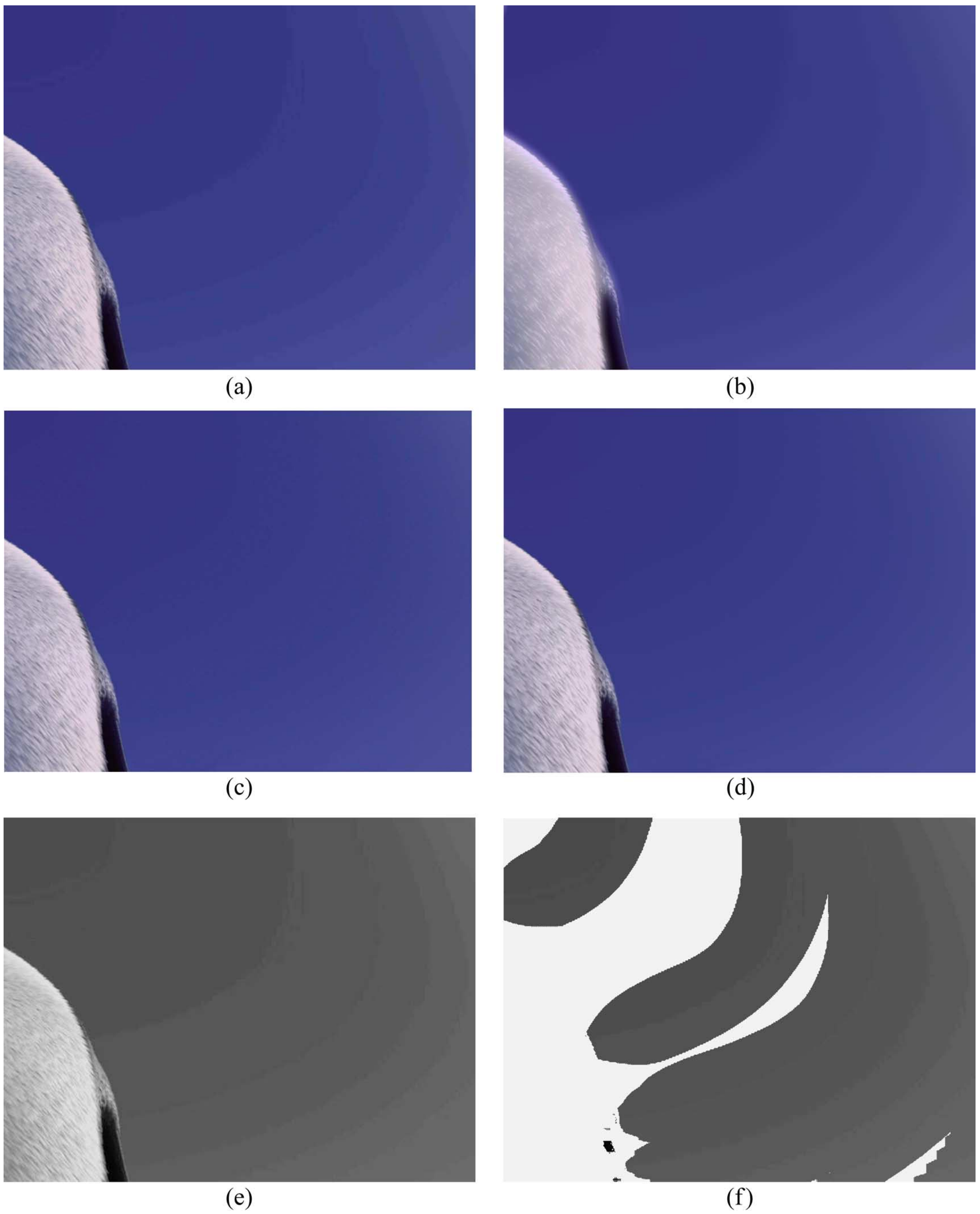

Fig. 5. (a) 6 bits-per-component (bpc) image with contour artifacts; (b) result of the "decontouring" method of Daly *et al.* [7] at 8 bpc; (c) result of the method proposed in Section III-B at 6 bpc; (d) result of the method proposed in Section IV at 8 bpc (bit depth extension); (e) the Y-component of the original 6 bpc image; and (f) the regions in the Y-component detected as constituting contour artifacts (same image as (e) but wherein regions *not* containing contour artifacts are masked out by white space). (Note that these images are best viewed in the electronic pdf version.)

sity). It also ensures that artifact-free regions, including small details (the white dots), mostly remain unaffected during the dithering process. Note that, in our implementation, $N_s(x,y)$ (in Section II) is a square neighborhood[7] centered at (x,y), and $T = 0.2$ in (2).

[7]At the borders of the image, $N_s(x,y)$ may not lie entirely inside the image. Ways of handling this problem include truncating the neighborhood to only the portion that lies inside the image, mirroring a band of border pixels, and ignoring a band of border pixels so that the neighborhood always stays inside the image.

In applications where the bit depth of the output can be higher than that of the input, the bit depth extension method (Section IV) may be applied. This helps us to suppress contour artifacts while introducing less perceivable noise than at lower bit depths. Fig. 4(e) shows the result after extending the bit depth from 6 to 8 bpp. It can be observed that the result at 8 bpp has less perceivable noise than the results at 6 bpp [see Fig. 4(c) and (d)].

Since contour artifacts arise from quantization of smooth gradients, the artifact reduction process should result in the rein-

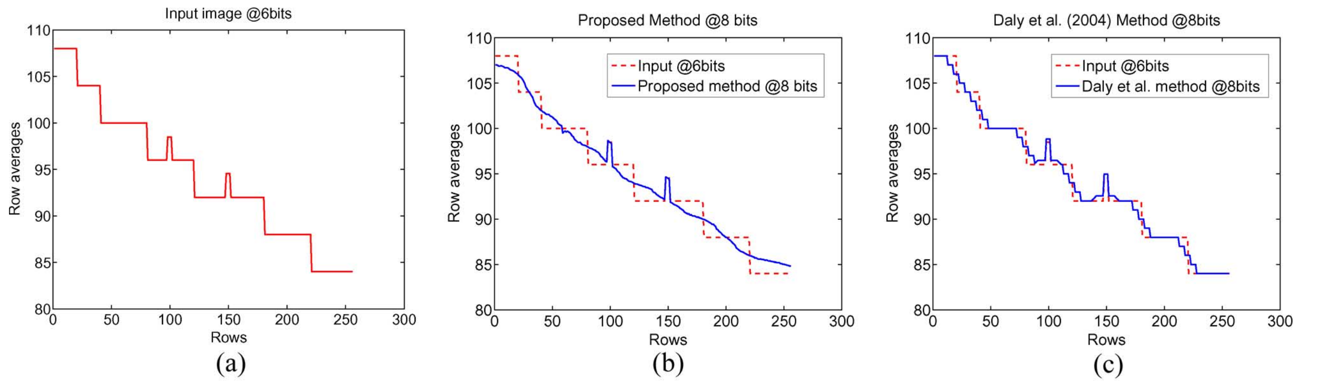

Fig. 6. Row averages of intensities for the image in Fig. 4(a) using (a) the input image; (b) the method proposed in Section IV; and (c) the method of Daly *et al.* [7]. The two intermittent peaks in the profile are due to the presence of the white dots in Fig. 4(a).

troduction of smooth gradients. We now compare the proposed method with that of Daly *et al.* in terms of the smoothness of the output intensity gradients. Fig. 6(a) plots the intensity average of each row of the 6 bpp input image shown in Fig. 4(a). The two intermediate peaks in the curves are due to the white dots in the input image. Fig. 6(b) shows in addition the row-wise intensity averages of the 8 bpp output image [shown in Fig. 4(e)] after applying the proposed method. Note that the staircase profile of the input image is smoothed out effectively by applying the proposed method. In comparison, Fig. 6(c) shows the row averages for the result obtained using the method of Daly *et al.* [shown in Fig. 4(b)]. The resulting image also has a staircase profile, although the "stairs" are of smaller scale than in the input. These observations are significant because the human visual system (HVS) cannot resolve individual pixel colors from a distance. Rather, it sees a local average of colors in the neighborhood of each pixel. By exploiting this averaging process in the HVS (as the proposed method does through dithering), we can actually see colors intermediate to those that can be represented by our digital code. In other words, our algorithm increases the effective color resolution of the image although the same set of code values are used.

Having illustrated the proposed approach using a synthetic grayscale example, let us now move on to a 6 bits-per-component (bpc) color image [shown in Fig. 5(a)] representing a portion of a frame of a high-definition animation movie. Quality loss due to contour artifacts resulting from color quantization is a major concern in the animation movie industry (since false contours are generally more visible in images with little or no noise). Fig. 5(a) contains a region of blue sky with severe contour artifacts because of the slowly varying color therein. It also contains an object with minute furry details. Ideally, we would like to remove the artifacts without affecting the furry details. Fig. 5(b) shows the decontouring result using the method of Daly *et al.* Once again, a 20×20 averaging filter is chosen for this algorithm since it seems to be the most effective scale for this image. Contour artifacts, although significantly reduced, are still visible on closer observation. More importantly, the furry details in the object are blurred and a thin halo is evident around the object. Thus, this method trades off the reduction of contour artifacts with the introduction of further artifacts and loss of detail. Fig. 5(c) shows the result of the contour artifact reduction method proposed in Section III-B at 6 bpc. Note that the contour artifacts have been reduced significantly and the furry details remain unaffected. Fig. 5(d) shows the 8 bpc result after applying the bit depth extension method of Section IV. This result retains the merits of that in Fig. 5(c), i.e., reduced artifacts and preservation of detail. In addition, it has less perceivable noise than the result in Fig. 5(c) as the dithering occurs at a higher bit depth.

Fig. 5(e) and (f) demonstrates the utility of the multiscale contour artifact detection process, described in Section II. As mentioned earlier, the contour artifact detection and reduction methods work on a per-color-component basis. The image in Fig. 5(a) is processed in the YUV color space. Fig. 5(e) shows the Y-component (luma) of this image. Fig. 5(f) shows the regions in the Y-component that have been detected as constituting contour artifacts. It is the same image as Fig. 5(e) but wherein regions *not* containing contour artifacts are masked out by white space. Note that most of the contour artifacts in this component are detected, even though they vary widely in scale. Note also that the region on the left with the furry texture is accurately classified as being free of contour artifacts. Thus, the multiscale contour artifact detection method enables reducing contour artifacts of a wide range of scales while leaving artifact-free regions unchanged. The detection method could, however, result in some false alarms (despite taking the precautionary measures described in Section II-A) such as the spurious blobs in the lower-left portion of Fig. 5(f). This happens in rare cases when an intended image pattern contains adjacent regions differing by ± 1 color value, and thus mimics a contour artifact. However, such patterns are normally unintended since they are nearly imperceivable in high bit-depth images used during content creation.

Fig. 7 shows more examples demonstrating the effectiveness of the proposed approach. For each row, the left-hand side shows the original 6 bpc image and the right-hand side shows the corresponding output image at the same bit depth. The example in the top row is cropped from a frame in an animation movie and the one in the bottom row is from a nonanimation movie. Observe that false contours are effectively masked and picture details largely remain unaffected. Regarding the second example, note that the noise normally present in nonanimation content is not always sufficient to prevent the occurrence of visible false contours during bit depth reduction.

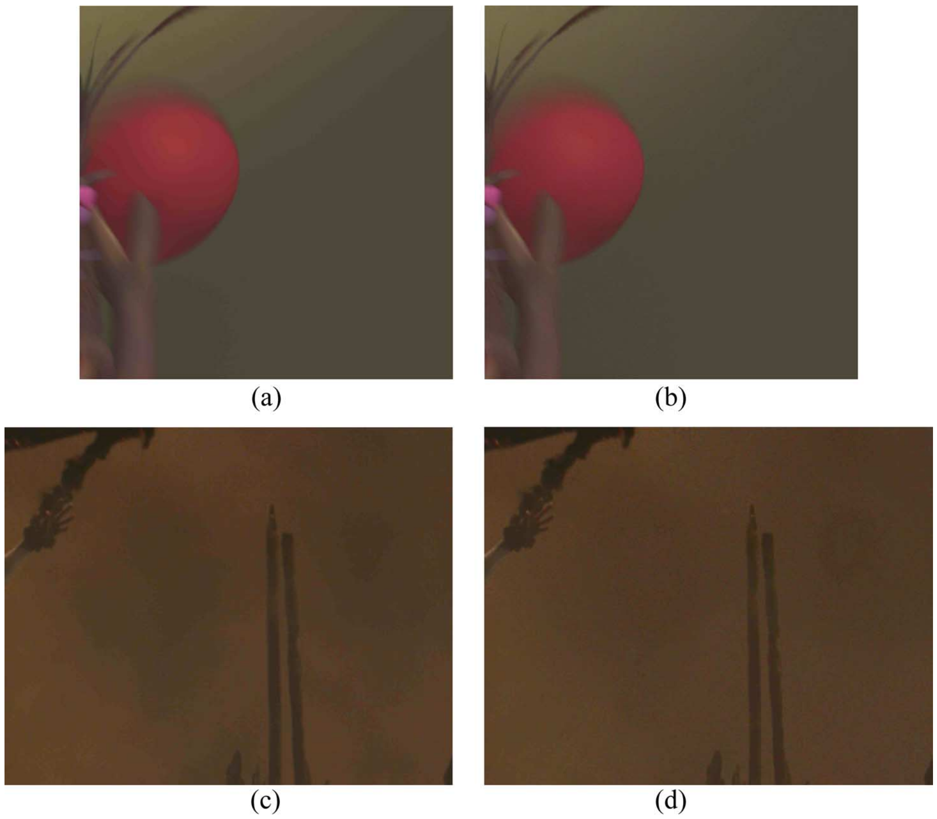

Fig. 7. More examples of contour artifact reduction using the method proposed in Section III-B. The left-hand side shows the original 6 bpc images and the right-hand side shows the corresponding output images at the same bit depth. The example in the top row is cropped from a frame in an animation movie and the one in the bottom row is from a nonanimation movie. (Note that these images are best viewed in the electronic pdf version.)

We conclude this section by providing run-time estimates for the proposed approach (using the dithering method described in Section III-B). The estimates are obtained by processing 1070 frames from two high-definition (1920×1080 pixel) color sequences on a PC with a 64-bit, 2.6 GHz AMD Opteron CPU. The average time-per-frame lies in the range 2.23 to 2.75 s, depending on the amount of contour artifacts detected. The lower bound corresponds to frames where no contour artifacts are detected (i.e., no dithering) and the upper bound corresponds to frames where artifacts are detected at all pixels (i.e., all pixels are dithered). Although current applications involve high-quality offline processing, the method could be feasible for real-time conversion and display applications since several ways of optimization are foreseen (e.g., parallelization as the method processes frames independently of each other).

VI. Conclusion

In this paper, we have proposed a method for reducing the visibility of contour artifacts, i.e., false contours arising from color quantization in digital images. The method comprises two steps, multiscale contour artifact detection and probabilistic dithering. The former enables the effective handling of contour artifacts of various scales. It also ensures that artifact-free regions and small details remain unaffected during the dithering process. The probabilistic dithering step breaks down false contours, reducing the visibility of contour artifacts. Unlike previous methods of contour artifact reduction, the proposed approach can provide an output image with the same bit depth as the input image or with higher bit depths. We have provided experimental results demonstrating the effectiveness of the proposed method and its advantages over a prominent previous method.

One issue worth noting is that dithering introduces some spatial noise in the picture and this may be undesirable when processing noise-free computer generated images. The proposed dithering method only changes the pixel values by ± 1 and, therefore, introduces minimal perceivable noise.[8] This low amount of noise is usually well worth the tradeoff as false contours are particularly easy to see in noise-free images.

In addition to spatial noisiness, there is the issue of temporal noisiness. Applying probabilistic dithering independently on each frame results in the sequence looking noisier due to the random temporal variation of colors. This noisy look is especially undesirable if the initial sequence contains little or no noise, as in the case of animation content. In order to address this issue, an interesting direction of future research is to enforce temporal consistency while processing a video sequence. The "moving" noise may be reduced by enforcing some form of interframe temporal consistency while applying probabilistic dithering.

References

[1] L. G. Roberts, "Picture coding using pseudo-random noise," *IRE Trans. Inf. Theory*, vol. IT-8, no. 2, pp. 145–154, Feb. 1962.

[8]This "noise" is sometimes close to imperceivable when the target bit depths are relatively high.

[2] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial grayscale," in *Proc. Soc. Information Display*, 1976, vol. 17, no. 2, pp. 75–77.
[3] V. Ostromoukhov, "A simple and efficient error-diffusion algorithm," in *Proc. SIGGRAPH ACM Computer Graphics, Annu. Conf. Ser.*, 2001, pp. 567–572.
[4] G. Joy and Z. Xiang, "Reducing false contours in quantized color images," *Comput. Graph.*, vol. 20, no. 2, pp. 231–242, 1996.
[5] J. W. Lee, B. R. Lim, R.-H. Park, J.-S. Kim, and W. Ahn, "Two-stage false contour detection using directional contrast and its application to adaptive false contour reduction," *IEEE Trans. Consum. Electron.*, vol. 52, no. 1, pp. 179–188, Feb. 2006.
[6] W. Ahn and J.-S. Kim, "Flat-region detection and false contour removal in the digital TV display," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2005, pp. 1338–1341.
[7] S. Daly and X. Feng, "Decontouring: Prevention and removal of false contour artifacts," in *Proc. SPIE*, 2004, vol. 5292, pp. 130–149.
[8] S. Daly, "Perceptual mechanisms in balancing the visibility of false contours and true details under bit-depth constraints," in *Proc. International Workshop on Image Media Quality and Its Applications (IMQA)*, 2008.
[9] Y. L. Lee, H. C. Kim, and H. W. Park, "Blocking effect reduction of JPEG images by signal adaptive filtering," *IEEE Trans. Image Process.*, vol. 7, no. 2, pp. 229–234, Feb. 1998.
[10] H. W. Park and Y. L. Lee, "A postprocessing method for reducing quantization effects in low bit-rate moving picture coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 161–171, Feb. 1999.

Sitaram Bhagavathy (S'98–M'06) received the B.E. degree in electronics and communication engineering in 1999 from the Karnataka Regional Engineering College (Surathkal), India, and the M.S. and Ph.D. degrees in electrical and computer engineering, in 2000 and 2005, respectively, from the University of California, Santa Barbara.

He is currently with the Signal Acquisition and Processing Group, Thomson Corporate Research, Princeton, NJ. His research interests are in the areas of image/video analysis and processing, computer vision, and pattern recognition.

Joan Llach (S'98–M'06) received the M.S. and Ph.D. degrees in electrical engineering from the Universitat Politècnica de Catalunya (UPC), Spain, in 1997 and 2003, respectively.

He was with Philips Research France from 2000 to 2001 and is now with Thomson Corporate Research, Princeton, NJ, where he manages the Signal Acquisition and Processing Group. His current research interests include 2-D and 3-D video analysis, processing, and compression, and high dynamic range and wide color gamut imagery.

Jiefu Zhai (M'05) received the B.E. degree in electrical engineering from Xi'an Jiaotong University, China, in 2001, and the M.S. degree in computer engineering from Institute of Computing Technology, Chinese Academy of Scienes.

He is currently with the Signal Acquisition and Processing Group, Thomson Corporate Research, Princeton, NJ. His research interests are in the areas of video processing and video compression.