Color Embedding Into Gray Images

Ricardo L. de Queiroz[1] and Karen M. Braun[2]
[1]Universidade de Brasilia, Brasilia, Brazil, queiroz@ieee.org
[2]Xerox Corporation, Webster, NY, US, kbraun@crt.xerox.com

***Abstract*—We present a reversible method to convert color graphics and pictures to gray images based on mapping colors to low-visibility high-frequency textures. From a monochrome textured image, the decoder can identify the textures and recover the color information. An image is textured by carrying a wavelet transform and replacing band-pass sub-bands by the chrominance images. The low-pass sub-band is the same as that of the luminance signal. The decoder performs a wavelet transform on the received gray image and recovers the chrominance channels. Registration problems are discussed and examples are presented.**

I. Introduction

COLOR documents are commonplace in contemporary offices and appear in a variety of forms. Documents are frequently prepared, stored, and displayed electronically, but they are also commonly printed and distributed as hardcopies. From brochures to technical papers, printed paper is still an important component of an office. When digital color documents are to be printed using a black-and-white printer or transmitted using a conventional black-and-white fax machine, we are faced with the problem of representing color images in black-and-white, while trying to retain the information conveyed in charts and pictures. Graphics, like pie charts, were likely prepared using very contrasting colors to enhance visibility. Once the color graphic is converted to monochrome, sometimes the contrasting colors are mapped to the same gray level and their visual difference vanishes. So, the first problem is how to convert colors to black and white such that different colors would look different on paper too, even if they have the same luminance component.

Beyond the above problem, we devised a color-to-gray mapping that is reversible; that is, given the monochrome image, or black and white printed paper produced with our method, we can recover the original colors.

II. From Color to Textured Gray

The problem with using the luminance component [1] as the gray image is that regions that have contrasting colors with similar luminance would be assigned similar output look the same. Figure 1 shows an example of a colorful map that might have different colors translated into similar shades of gray, thus obfuscating the borders between countries.

An alternative is to compute the colors in the graphic (typically a small number of distinct colors) and to assign different levels of gray to all neighboring colors. This approach may not work for complex graphics. Another approach is to map colors to textures. One can control

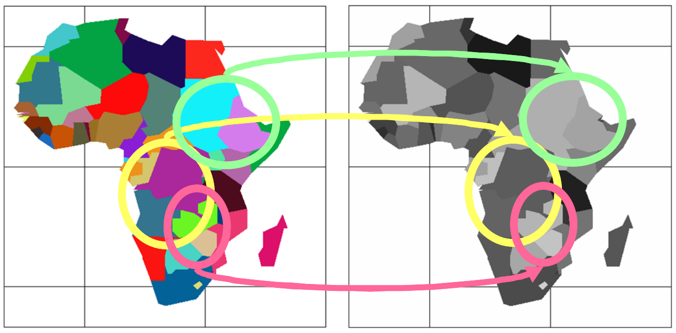
Fig. 1. The mapping from color to monochrome (gray) images.

halftone dots or patterns as a function of the colors (e.g. as a function of hue and saturation). Hence, regions of different colors with similar luminance will look different after mapping because they would have different textures.

Our method maps colors to texture. However, instead of having a dictionary (or palette) of textures and colors, it produces a continuum of textures that naturally switch between patterns without causing visual artifacts. The method works as follows:

1) The color image, assumed to be in any RGB color space is transformed into Y, C_b, C_r planes using popular compression RGB-YC_bC_r linear color transformations [2]-[3], even though a color space like CIELab [1] would work equally well.
2) Using one level of the wavelet transform [4], the luminance image Y is divided into 4 sub-bands: $Y \rightarrow (S_l, S_h, S_v, S_d)$, corresponding to the low-pass, vertical, horizontal and diagonal (high-pass in both directions) sub-bands, respectively. Using decimated filter banks, the dimensions of S_l, S_h, S_v, S_d are half of those of Y in each direction. Oversampled filter banks and wavelets would also work.
3) The planes C_b and C_r are spatially reduced by a factor of 2 in each direction.
4) S_h,is replaced by C_b and S_v,is replaced by C_r.
5) An inverse wavelet (sub-band) transform [4] is carried to recompose the monochrome image as $(S_l, C_b, C_r, S_d) \rightarrow Y'$
6) Image Y' is the resulting gray image and may be printed, which often includes scaling and halftoning.

The process is illustrated in Fig. 2. Both the high-pass bands and chrominance signals adapt well to scene object contours, making the texture pattern changes appear to be natural. Apart from being based on wavelets, the novelty in this method lies on three other key aspects: (i) the texture differentiation is applied to the gray image and not directly to the halftone image; (ii) its smooth and natural color blending is suitable to both graphics and pictures; and, most important, (iii) it is reversible, enabling retrieval of the colors from the textured image.

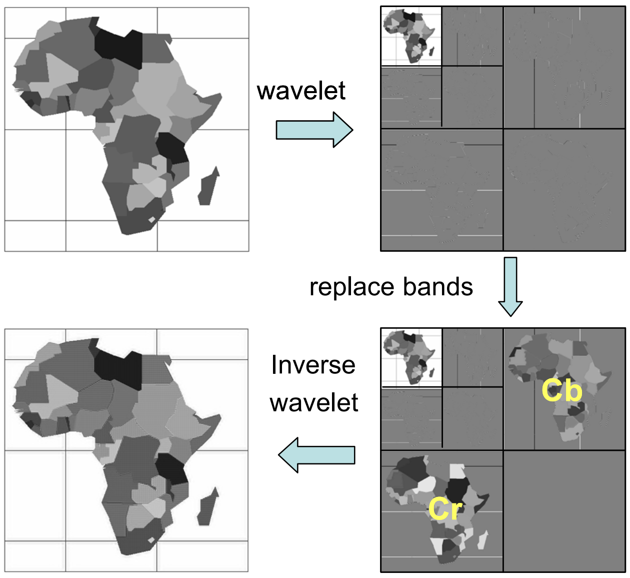

Fig. 2. The mapping from color to monochrome (gray) images. If Moiré is visible please change viewing resolution.

III. Recovering Color

One nice feature of the proposed embedding method is the ability to recover the color from the gray (textured) image. For that, we just need to reverse all steps in the color-to-gray mapping. The proposed method for embedding and retrieving color into and from a gray image is theoretically sound but faces some practical obstacles, when one halftones, prints and scans the gray image. First, after halftoning and perhaps printing and scanning back the image, the decoder (which maps the gray image back to color) needs to filter out the halftone and to scale the image to its right size. Second, printing may warp the image, stretching the paper. Scanning may not be properly aligned, causing the recovered gray image to be a warped version of the one before printing. Results can be catastrophic. Figure 5 depicts the situation where a vertical texture pattern (which should not produce any vertical high frequency coefficients) is rotated by as little as half a degree. Such a small rotation will cause low frequency vertical patterns and distort the horizontal sub-bands. An example is shown in Fig. 3. Third, the image needs to have perfect scanning registration. Any shift in the texture image may cause major color shifts as in the example in Fig. 4. Fourth and finally, the sharp texture we apply is blurred as a result of the printing process which translates to desaturation of the output image. The output image saturation can be boosted to account for this.

To deal with these issues, we have to scale the image before halftoning and printing enough to ensure the gray texture patterns will survive printing, scanning, and filtering. Also, we de-warp and scale back the scanned image before processing. To do that, we detect corners of the scanned image and perform an affine transformation to make the scanned rectangle fit a specified image dimension. The inconvenience is that the decoder must know the image size and scaling. This can be solved by only allowing a small number of image dimensions and estimating which image size was used. Most important of all, we changed the way we embed the color information into the subbands.

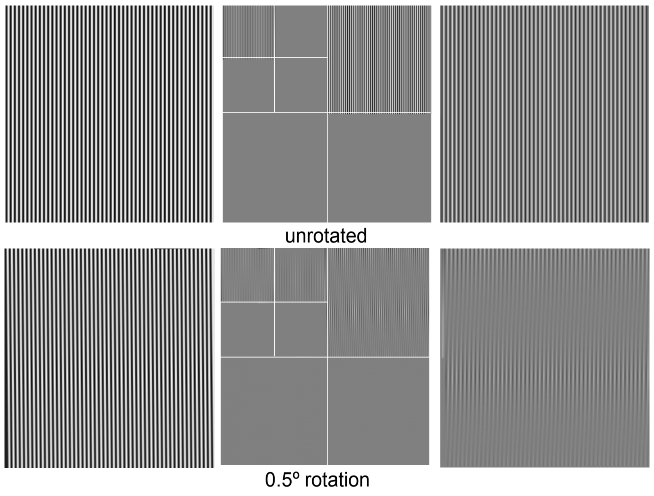

Fig. 3. The effect of even the slightest rotation. Vertical texture (left) produces virtually no horizontal patterns in wavelet domain (center). A zoom of the horizontal sub-band is shown (right). After 0.5° rotation, low-frequency vertical patterns appear. Similar results are repeated for the rotated texture in the bottom row.

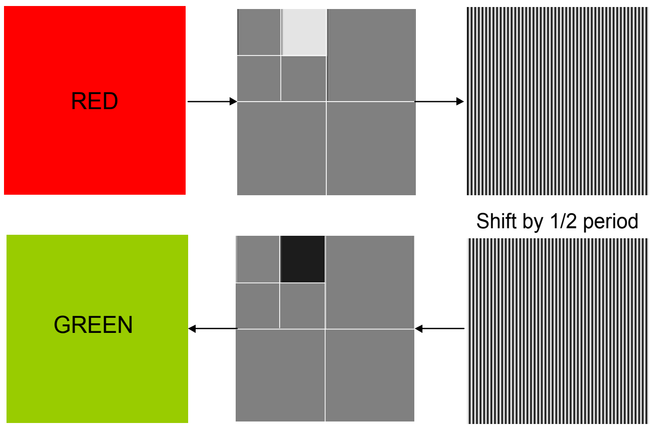

Fig.4. Opposite colors produce inverted textures. Small shifts in the image can cause the textures to be inverted and lead to large color recovery errors.

In order to get more robust color embedding against decoding opposite colors caused by a small image shift, we divided the chrominance information into 4 planes. C_b is divided into two planes: C_b+ and C_b-. In C_b+ we reproduce the pixels of C_b which are greater than 0, i.e. $C_b+ = (C_b>0)$. The remaining pixels are set to zero. Same thing happens for C_b-. In C_b- we reproduce the pixels of C_b which less than 0, i.e. $C_b- = (C_b<0)$. The remaining pixels are set to zero. The same arrangement is made for C_r. Note that $C_b=(C_b+) + (C_b-)$ and $C_r=(C_r+) + (C_r-)$. The reason to create positive- and negative-valued chrominance planes is to avoid completely the color inversion problem depicted in Fig. 6. If a sub-band is supposed to have only positive values and we obtained negative ones, then it is a sign of texture inversion and we should use the absolute value of sub-bands, i.e.

$$C_b = |C_b+| - |C_b-| \quad and \quad C_r = |C_r+| - |C_r-| \,.$$

As a result, we have 4 images to embed: C_b+, C_b-, C_r+, and C_r-. If we do a 2-level wavelet transform, the image plane Y is transformed into $Y \rightarrow (S_l, S_{h1}, S_{v1}, S_{d1}, S_{h2}, S_{v2}, S_{d2})$, where the level-2 sub-bands are the higher-frequency bands. Then, band replacement occurs as follows:

$S_{d1} \leftarrow C_b-$; $S_{h2} \leftarrow C_r+$; $S_{v2} \leftarrow C_b+$; $S_{d2} \leftarrow C_r-$.

Since S_{d1} has lower resolution than the others, one has to reduce C_b-, further to ¼ of the resolution in each dimension, compared to the original Y plane. The color embedding scheme is illustrated in Fig. 5 while the recovery process is illustrated in Fig. 6. The embedding and recovery steps are:

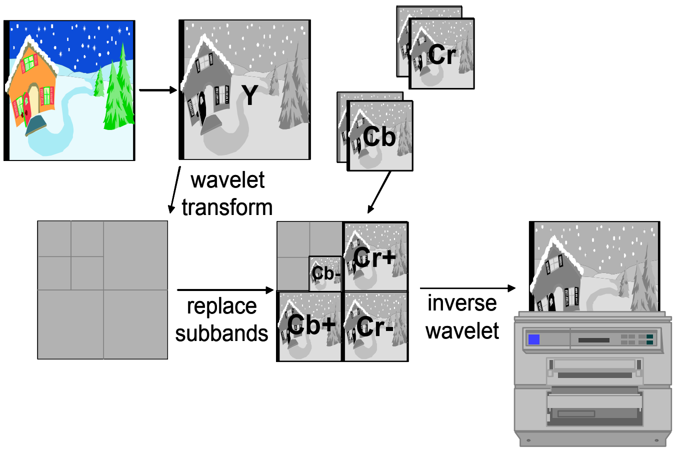

Fig. 5. The mapping from color to monochrome (gray) images through color embedding into textured gray images. We employ sub-band replacement by absolute-valued chrominance planes.

<u>Color embedding</u>:

1) Convert image from RGB into Y, C_b, C_r (or CIELab).

2) Use a two-level wavelet transform on Y , so that Y is divided into 7 sub-bands: $Y \rightarrow (S_l, S_{h1}, S_{v1}, S_{d1}, S_{h2}, S_{v2}, S_{d2})$.

3) Reduce C_b and C_r by ½, construct C_b+, C_b-, C_r+, C_r-, and reduce C_b- further to ¼ of its original size.

4) Replace sub-bands

$S_{d1} \leftarrow C_b-$; $S_{h2} \leftarrow C_r+$; $S_{v2} \leftarrow C_b+$; $S_{d2} \leftarrow C_r-$.

5) Take inverse wavelet transform to obtain the textured gray image, i.e. $(S_l, S_{h1}, S_{v1}, C_b-, C_r+, C_b+, C_r-) \rightarrow Y'$.

6) Scale, halftone, and print or transmit Y' (the gray image).

<u>Color recovery</u>:

1) Read or scan the gray textured image.

2) Determine image dimensions., identify corners and carry an affine transform to de-warp the gray image.

3) Reduce image to the correct resolution.

4) Use a wavelet transform to convert the gray image into sub-bands $Y' \rightarrow (S_l, S_{h1}, S_{v1}, S_{d1}, S_{h2}, S_{v2}, S_{d2})$.

5) Interpolate S_{d1}, doubling its resolution.

6) Make $C_b = |S_{v2}| - |S_{d1}|$, and $C_r = |S_{h2}| - |S_{d2}|$.

7) Interpolate C_b and Cr, doubling their resolutions.

8) Remove the embedded sub-bands, i.e. set $S_{d1} = S_{h2} = S_{v2} = S_{d2} = 0$, and take the inverse wavelet transform to find Y as $(S_l, S_{h1}, S_{v1}, 0, 0, 0, 0) \rightarrow Y$.

9) Convert the Y, C_b, C_r planes back to RGB.

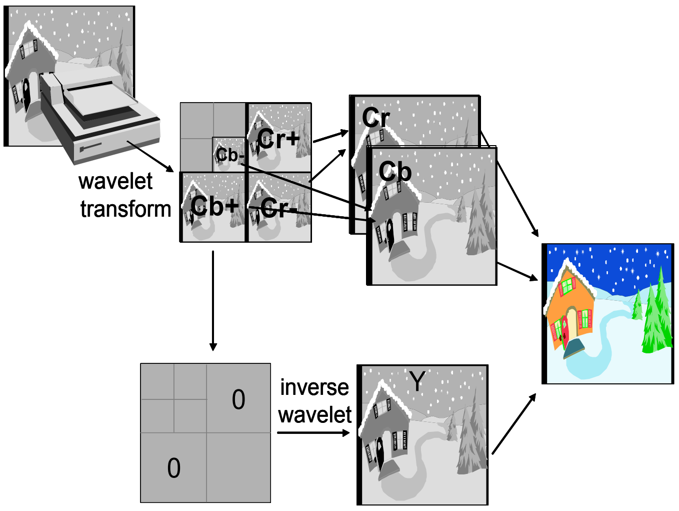

Fig. 6. Recovering color from a textured gray image. The embedded sub-bands are recovered to form the chrominance planes and zeroed before inverse transform. The YC_bC_r data is then converted back into RGB.

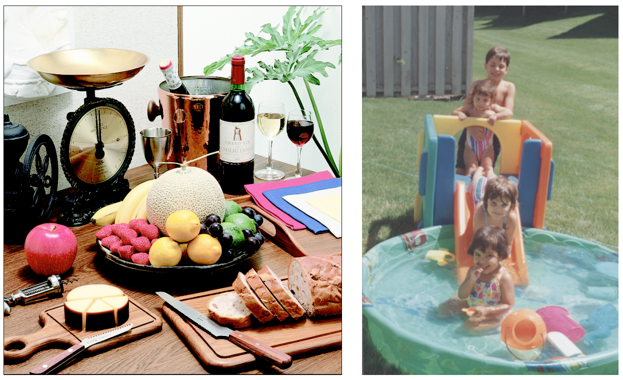

Fig. 7. Two example color images: "wine" (764x832 pixels) and "kids" (488x340 pixels).

IV. Results

We have tested the algorithm with and without going through the printing and scanning cycle. The great difficulty with printing and scanning is the non-uniform stretching and rotation that might occur to the image after it is printed and then scanned. This is a hard registration problem that is common to many machine reading systems and is beyond the scope of this paper. In some cases like when sending the halftoned image via standard black and white faxes, registration is not an issue.

Fig. 7 shows two typical color images, picked randomly among the many we tested. The textured images are shown in Fig. 8. The high-frequency textures have low visibility and blend well with the image. The textured image is often spatially scaled up by an integer factor K (e.g. K=4) in each direction, before printing, to ensure the texture will survive the printing and scanning process. In the simulation mode, the scaled images are halftoned using standard error diffusion or any other method, then reduced by averaging KxK binary pixels to recompose a gray image. The recovered "kids" image for different scaling factors is shown in Fig. 9. Note how some colors are de-saturated. Nevertheless the colors seem to have the right hue and, on average, are approximately correct. The larger *K*, the better the reconstruction. If we compute the peak signal-to-noise ratio (PSNR) between input and recovered RGB images, for image "kids", the results will be for example 13.7dB for *K*=1, 25.6dB for *K*=4 and 28.7dB for *K*=10. Figure 10 shows the reconstructed image "wine" using a scaling factor *K*=4.

The same image was printed using a standard 600 dpi laser printer with *K*=4 and scanned using a 1200 dpi scanner. A portion of this scanned image is shown in Fig. 13. After applying the affine transformations to de-warp and to de-rotate the gray textured image, the image after color recovery was sharpened and color enhanced yielding the image in Fig. 12. Taking into account all non-linear and unpredictable variables derived from the physical processes involved, we

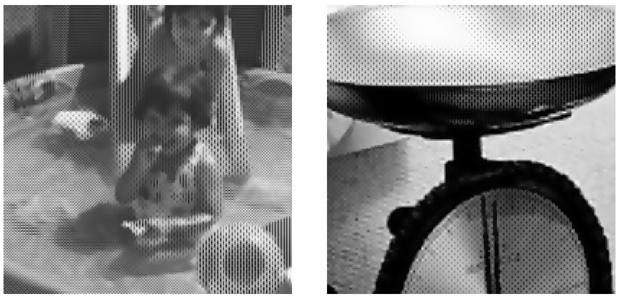

Fig. 8. Enlarged portions of textured images "kids" and "wine". Without enlargement (and without Acrobat's re-sampling) the texture is not noticeable.

consider images like those in Fig. 12 to be excellent results.

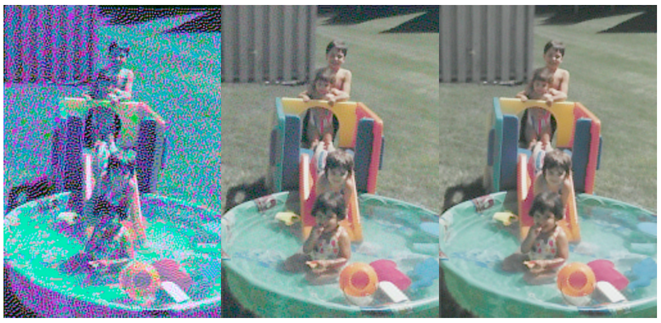

Fig. 9. Image with color recovery after error diffusion halftoning and scaling. From left to right, *K*=1, *K*=4 and *K*=8.

V. Conclusions

We have presented a method to convert color images to gray that is invertible and allows easy distinction of colors with similar luminance values. The highlight of this paper is the fact that the color to gray mapping is invertible by mapping color to textures which can be later decoded and converted back to color. We are unaware of any other attempt to do so. The method is based on wavelet transforms and on replacing sub-bands by chrominance planes.

Registration and geometric distortions are still problems, except for fax transmission. Our next research step is to try shift invariant, complex wavelets, to test whether they would be more robust against geometric image distortions caused by the printing and scanning processes.

References

[1] R. W. G. Hunt, *The Reproduction of Color*, Fountain Press, Toolworth, England, 2000.

[2] W. B. Pennebaker and J. L Mitchell., *JPEG: Still Image Compression Standard*, Van Nostrand Reinhold, NY, 1993.

[3] R. L. de Queiroz, *Compression of Color Images* , in *The Handbook on Transforms and Data Compression,* edited by G. Sharma, CRC Press, 2002.

[4] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge, Welesley, MA, 1996.

Textured Images in Figures 2, 3, 4, 8 will likely be displayed incorrectly by a combination of resampling artifacts caused by the conversion and display of PDF. Since they are crucial to the paper, please take a look at the images in http://image.unb.br/queiroz/papers/textured-figures.doc

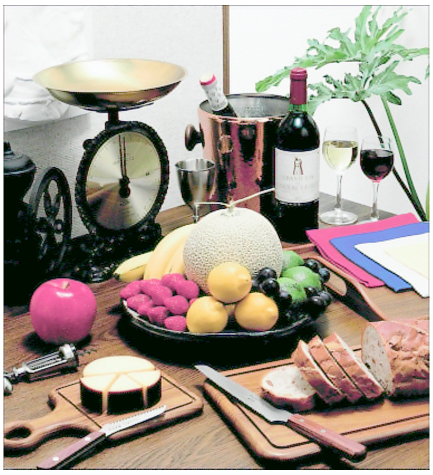

Fig. 10. Image "wine" recovered after haftoning and scaling with *K*=4.

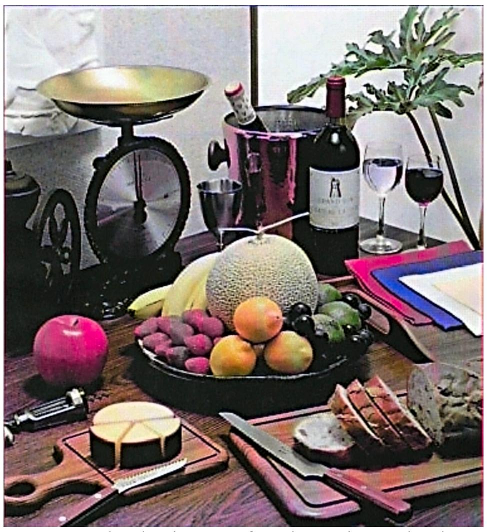

Fig. 11. Reconstructed color image, after sharpening and color enhancement (saturation) .