A COLOR DITHERING ALGORITHM

Radu-Lucian Lupşa, Dana Lupşa

Abstract

This paper presents a very general method for color dithering – it is not restricted as far as the number or the color of inks is concerned.

Keywords: image rendering

1 Introduction

When printing an image, it is necessary to produce the colors in the image from a small number of inks. The colors are produced by alternating dots of diferent inks, so that the average color in a small region around each pixel in the rendered image approaches the desired value (the value in the original image).

There are four distinct uses of dithering:

- offset printing
- printing on an inkjet or laser printer
- displaying of an image on a fixed palette display
- displaying of an image on an application defined palette display

This article adresses only the printer case, and the fixed palette display as a special case.

There are various dithering algorithms available for monochrome images. Most of them are of error-diffusion type, that is, the pixels are taken in a predefined order,forming a path throughout the image, and the output pixel values are generated, while maintaining the error value (the difference between the sum of the pixel values seen so far in the original image and the sum of the generated output pixel values). That error is distributed to some neighbour pixels (and each error-diffusion algorithm comes with a scheme). The advantage is simplicity and speed; the disadvantage is that there are, for each algorithm, some "bad gary levels", that is, levels that generates some annoing repetitive patterns — such as stright vertical or sloped line, or other "artifacts".

2 Quad-tree error diffusion

2.1 Quad-trees

Let n be a positive integer (for most of our tests, $n = 3$), and let us suppose the image sizes are multiples of 2^n. Now, let's consider the division of the image into squares of 2^n by 2^n, and let's call it the *level 0 division.* Also, let us associate a quad tree to each such level 0 division square, with the root associated to the level 0 square.

Then, we divide each level 0 square into four squares. and we associate them with the sons of the corresponding root node in the corresponding

tree. The resulting squares will be the *level 1 division squares.* We continue the procedure until level n, where each division square will consist of exactly one pixel.

Now, for each division square, regardless the level, we can define an *average color* as being the average of the pixel values of the pixels in that square. We have therefore average values for the original image, and also average values for the dithered image we are constructing. We can also define a distance (or error) between the average value in the constructed image and the corresponding square in the original image.

Our method is trying to achieve, for each square, an error that is less than a certain threshold; the threshold will depend only on the level of the square.

2.2 Quad-tree error-diffusion for orthogonal colorspace

Let's suppose now that each pixel can have one of the colors: Black (0,0,0), Red (1,0,0), Green (0,1,0), Blue (0,0,1), Cyan (0,1,1), Yellow (1,1,0), Magenta (1,0,1), and White (1,1,1). So, each pixel can have any color that have integer components.

As a level k division square is $2^{n-k} \times 2^{n-k}$ pixels, its averaga color can be any color that is multiple of $\frac{1}{4^{n-k}}$. Moreover, the work on the three color components can be done independently.

Now, our algorithm takes each level 0 division square of the original image and, for each color component, computes that component for the pixels in the corresponding square of the constructed image. This is achieved in three steps:

1. First, compute the original image's average value in each node of the quad tree. This is performed bottom-up, from the leaves to the root.

2. Next, if the average value v_0 in the original image is a multiple of $\frac{1}{4^n}$, then let the average value u_0 of the root for the constructed image take the same value ($u_0 = v_0$). Otherwise, let

$$v_0' = \frac{[4^n v_0]}{4^n}$$

   (the square brakets denote the integral part) and we put the average value the average value of the root for the constructed image be $u_0 = v_0' + X$, where X is a random value that can be 0 or 1 with the probability of being 1 equal to $4^n(v_0 - v_0')$.

3. For each non-leaf node of the tree associated with the constructed image, we compute the average value of its sons as follows:

   Let k be the level of the current, non-leaf, node (the sons are then on level $k+1$, and $0 \leq k < n$). Then, let v_0 be the average value of the current node for the original image, $v_0' = \frac{[4^{n-k} v_0]}{4^{n-k}}$, u_0 the average value of the curent node for the constructed image ($u_0 = v_o'$ or $u_0 = v_0' + \frac{1}{4^{n-k}}$), v_1, v_2, v_3, and v_4 the average values of the sons of the current node in the original image, and we shall construct

the average values u_1, u_2, u_3, and u_4 of the sons in the constructed image.

Now, we put, for $i = \overline{1,4}$:

$$v'_i = \frac{\left[4^{n-k-1} v_i\right]}{4^{n-k-1}},$$

and,

$$d_i = v_i - v'_i.$$

It is easy to see that, if $u_0 = v'_0$ or $u_0 = v'_0 + \frac{1}{4^{n-k}}$, then u_0 lies between $\sum_{i=1}^{4} v'_i$ and u_0 lies between $\sum_{i=1}^{4} v'_i + 4\frac{1}{4^{n-k-1}}$. Therefore, for each $i = \overline{1,4}$, we can put either $u_i = v'_i$ or $u_i = v'_i + \frac{1}{4^{n-k-1}}$.

So, the ideea is, the same as for the root node, to put $u_i = v'_i + X_i \frac{1}{4^{n-k-1}}$, where X_i is a random variable that can take the values 0 and 1, with the probability to be 1 of $\frac{d_i}{d_1+d_2+d_3+d_4}$. Unfortunately, it is possible that $\frac{d_i}{d_1+d_2+d_3+d_4>1.}$ So, we just sort d_i in decreasing order, and if $\frac{d_i}{d_1+d_2+d_3+d_4>1}$ then we put $X_i = 1$.

The algorithm guarantees, for each division square, a difference between the original image average value and the dithered image average color of at most the contribution of one pixel. Also, there are no artifacts as long as the random number source is good enough.

3 Methods for non-orthogonal colorspace

In the previous section we considered that each pixel can take any color with color components of 0 and 1. For a real color printer, this condition cannot be met but within rather coarse approximation. A real yellow ink, for instance, can never reflect as much red and green light as the white paper, nor can it absorb all the blue component. Also, combining two inks does not result in a reflection coefficient exactly equal to the product of the reflection coefficient of the two component inks. See [1] and [2] for a complete light reflection model.

Therefore, we must first measure the reflection coefficients of the paper and of all available inks and combinations of inks. Let C be the set of colors of the white paper and all the inks and combinations of inks. (in fact, each element of C is the tuple (c_r, c_g, c_b) of the reflection coeficients for the red, green and blue light).

Now, C is the set of achieveable colors for one pixel.

The dithering algorithm takes again each level 0 square independently:

1. For each node, from leaves to the root, compute the average color and the set S of admisible achieveable average colors. The set S for a leaf node is a subset of C of the first colors from S in decreasing distance from the pixel color in the original image; also we require that that distance be below a given threshold. For non-leaf nodes, the set S is a subset of $S_1 + S_2 + S_3 + S_4$, where S_1, S_2, S_3, and S_4 are the sets of admisible achievable colors for the four sons of the current node.

2. For the root node, choose the best admisible achievable color (from the associated S set).
3. For all other nodes, pick the color in S that participated in the generation of the color of the parent node.

4 Conclusions and future work

The quad-tree error-diffusion algorithm is a fast dithering algorithm giving a good-contrast image with no artifacts.

The method for non-orthogonal colorspace gives an acceptable dithered image for very small color set. However, it is slow (due to the exhaustive search in $S_1 + S_2 + S_3 + S_4$ which can have a few dozens to a few hundreds elements) and creates artifacts due to the deterministic approach. The expected improuvment should be to combine somehow the randomized error-diffusion with the second method.

References

[1] Joanna L. Power, Brad S. West, Eric J. Stollniz, David H. Salesin Reproducing Color Images Using Duotones *Proceedings of SIGGRAPH 96, p. 237–248, 1996*

[2] Eric J. Stollniz, Victor Ostromoukhov, David H. Salesin Reproducing Color Images Using Custom Inks *Proceedings of SIGGRAPH 98, p. 267–274, 1998*