Color Dithering with n-Best Algorithm

Kjell Lemström, Jorma Tarhio
University of Helsinki
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki
Finland
{klemstro,tarhio}@cs.Helsinki.fi

Tapio Takala
Helsinki University of Technology
Department of Computer Science
Otakaari 1
FIN-02150 Espoo
Finland
tta@cs.hut.fi

Abstract

One of the interesting problems of digital image processing is how much the color table can be reduced without any detectable difference in the quality of an image. The process of reducing the color information is known as quantizing. Often visible artifacts, such as false contours, appear in a quantized image. Digital halftoning or spatial dithering is a process which tries to reduce the visibility of quantization errors. A new method, called n-best, is presented for dithering color images. This method adds noise that is proportional to the quantization error at each pixel. The n-best algorithm turns out to be a good compromise between the efficiency and the quality of the produced image.

Keywords: digital halftoning, dithering, quantization

1 Introduction

Quantizing is an interesting problem in the area of computer graphics from both theoretical and practical point of view. The aim of quantizing is to decrease the number of colors without any disturbing effect to human eye. Quantization is necessary in practice as many hardware devices have an upper limit for the number of colors of an image. Quantizing algorithms can be static or adaptive. A static algorithm takes a predefined set of colors for the quantized image, without any information of the image to be quantized. An adaptive algorithm is more sophisticated. It builds a color table based on the information of the original image. Some adaptive quantization algorithms are presented in [3, 4, 14].

Some visible errors can appear in a quantized image. The process of minimizing these errors is called digital halftoning or spatial dithering. The idea is to change the quantized color of some pixels so that the errors are less visible. This is possible because human eye integrates the color information of several picture elements (pixels).

There are three basic methods for dithering. One is to add noise to the quantized image. The added noise should be controlled in some specific way so that the quality of the picture stays good. The other method is to use a dithering matrix (or filter). The latter is known as *ordered dither*[1]. The algorithms of ordered dither are usually fast but they produce regular patterns that are visible. Ordered dither algorithms are presented in articles [1, 5, 7]. A third method is to

[1] The term has a historical background; the method is ordered rather than random.

divide the quantizing error of each pixel to its neighborhood. This process is called *error diffusion*. Error diffusion algorithms [2, 5] usually produce good results but they are not so fast. There are also algorithms [6, 13] that are variations of the basic methods.

A new algorithm based on adding noise is presented. The idea is to search for every input color of the original image the nearest candidates from the color table produced by the quantization process. The algorithm selects one of these candidates to the final image to represent an input color. This is realized in such a way that the closer a candidate lies to the input color in the color space the greater is its probability to become chosen as a representative to the final image.

Problem setting. The input data is a separation of red, green and blue values of a digitized image. This kind of an image is known as an *rgb* image. In a typical format, each of these three color components is represented with eight bits which gives more than 16 million different colors. The term *original image* is used for the images of this type. The *final image* is the image which is quantized (and dithered).

The quantization produces a *color table* which contains colors that are somehow typical of the original image. Only one value of eight bits referring to this color (lookup) table is stored for each pixel in the *final image*.

Let set A contain all the different color values c of the original image. *Input color* $c \in A$ is a 24 bit rgb value in the color space. The size of set A is denoted by $|A|$. Note the difference between a pixel and an input color: $p = (x, y)$ is the spatial coordinate of pixel p and $c = f(x, y) = f(p)$ is the color value of p.

Set B contains the *candidate* values selected by the quantizer. From B a *representative* is chosen for every pixel of the image.

2 Quantization

Quantization is the process of assigning representative values to a set of input colors. Quantization can be presented as a mapping from set A to set B:

$$\mathcal{Q} : A \rightarrow B, \text{ where } |A| \geq |B|, \text{ but not necessarily } B \subseteq A \text{ holds.} \tag{1}$$

A grayscale image can be quantized to a bilevel image by just setting threshold value I_{th} for the intensity function of the image so that intensities $I \leq I_{th}$ represent black and intensities $I > I_{th}$ white.

Color image quantization contains two important stages. The first one is selecting a set of colors of the image to represent the color gamut. The latter is to define a mapping from the color space to these representative values. In the literature several algorithms [3, 4, 14] are presented for this problem. We concentrate on dithering color images. Our new dithering algorithm applies only one quantization algorithm due to efficiency. We chose to use Heckbert's *median cut algorithm* [4], but other quantization algorithms are also appropriate, they just have minor quality and efficiency differences.

Median cut algorithm. The idea of Heckbert's algorithm [4] is to divide the color space into $|B|$ cells. One color is chosen to be a representative for each cell.

The first step of quantization is to count the frequencies of input colors $c \in A$. To make this computation faster, only five most significant bits are taken in account. This *prequantization* gives us efficiency at the expense of the quality. However, the loss of information is not significant because there will be much less colors after the quantization process.

The next step is to divide the color space into three-dimensional cells. First there is only one cell containing all the input colors of the image. We select for the cell the dimension which is the widest (the maximal difference between the maximum and minimum value of one of the three rgb dimensions). The cell is divided by a plane into two cells at the point of the median of the widest dimension. Then we select the cell with the widest dimension among all the cells and divide it in the same way. This procedure is repeated until we have $|B|$ cells.

For each cell we select representative q by counting the weighted mean of the input colors that lie in the subspace defined by the cell. Representative q is stored to the color table and the index of q is written to the cell.

When the color table is created, a mapping from the input colors to the color table has to be defined. For each different input color c_i the closest possible $q \in B$ is selected as representative r. Note that values r and c_i do not necessarily lie in the same cell, but they have to lie in adjacent cells.

The final image is created by applying the following procedure: The original image is read again and the color value of each pixel is prequantized. This prequantized input color lies in one of the cells. We have to find the closest of the representatives of this cell and the adjacent cells, which color will be the final color of the pixel.

3 Dithering

In the past several dithering algorithms have been created for reducing the visible quantization errors and for making an illusion of more colors than there really are. The simplest idea is to add *white noise* to the picture. White noise is uniformly distributed over the whole power spectrum. The effect of this simple idea is unpleasant because the noise is disturbing. Ulichney [10, 11] found out that the noise should be at the high frequencies of the power spectrum and he presented a model called *blue noise*.

Ordered dither algorithms add pseudorandom noise according to a deterministic sequence. This sequence is defined by a filter. One filter used is shown in Figure 1.

$$F^4_{URT} = \frac{1}{16}\begin{bmatrix} 0 & 14 & 3 & 13 \\ 11 & 5 & 8 & 6 \\ 12 & 2 & 15 & 1 \\ 7 & 9 & 4 & 10 \end{bmatrix}$$

Figure 1: *A* 4×4 *ordered dither matrix.*

The idea is to carry the filter over the picture. The elements of the filter (see Figure 1) are threshold values to which the intensities are compared. Effectively this is the same as subtracting the filter values from the image and then quantizing with a constant threshold. When one region of the image is processed, the $n \times n$ filter is shifted n positions right or down. In fact, the threshold value for each pixel p can be calculated from equation $T_{xy} = F^n(x \bmod n, y \bmod n)$, where x and y are the coordinates of pixel p.

A different method is to diffuse the quantization error to the neighborhood of the pixel. In this approach, both the local and global intensities remain the same. Floyd and Steinberg [2] present an error diffusion algorithm of this kind. The idea of the Floyd-Steinberg algorithm (FS for short) is to divide the error to those neighbor pixels that are not yet handled in the scanline order. The proportions of the error forwarded to the neighbors are shown in Figure 2.

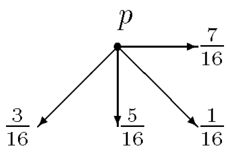

Figure 2: *The Floyd-Steinberg algorithm diffuses the quantization error to the neighborhood on the fractions showed in the figure.*

The disadvantage of the FS algorithm is that the error cannot be handled locally: error is propagated to the right and bottom edges of an image. When a small area is changed the whole picture has to be processed again. Knuth [6] presents an algorithm that diffuse the error in distinct 8×8 pixel regions. The error of each region is accumulated to two isolated pixels and therefore it cannot be distinguished.

In the literature other ideas are also presented to reduce the artifacts produced by traditional algorithms. It can be done by changing the order in which the pixels are handled [13] or simply by rotating the ordered dither matrix [7]. Our new algorithm adds pseudorandom noise to the image. It is called the n-best algorithm.

4 The n-Best Algorithm

Most of the dithering algorithms are applicable to greyscale images as well as color images. The algorithm to be presented, called *n-best*, is specially designed for color images. The algorithm creates a mapping from one input color to a set of n candidates.

The idea is straight-forward: For each pixel, colors are chosen from a set of candidates which lie near the input color in the rgb space. Probabilities are assigned to these candidates such that the probability is inversely proportional to the distance from the input color. When a pixel is to be handled, its color value c_i is read and the representative is chosen from set S_i randomly.

Because the representative is chosen from the n nearest candidates, the method is called *n-best*. If we choose $n = 1$, the image is only quantized.

An advantage of algorithms adding noise randomly is that they do not produce regular patterns, but usually the granularity is, however, disturbing. With the n-best algorithm the granularity can be decreased because the noise can be directed to areas where quantization errors occur.

4.1 Assigning the Probabilities

Let us call candidates and input colors as rgb values. Let $\mathcal{D}(x, y)$ be the distance between two rgb values x and y in the rgb space. Set S_i is created for every input color c_i so that S_i contains one or n candidates q_{ik}. When $q_{i1} = c_i$, set S_i contains only q_{i1} (remember that c_i is prequantized before dithering), otherwise S_i contains n candidates. Every q_{ik} satisfies inequality:

$$\mathcal{D}(q_{ik}, c_i) \leq \mathcal{D}(q', c_i), \; \forall q_{ik} \in S_i \;\wedge\; \forall q' \in B \setminus S_i. \tag{2}$$

In other words, S_i contains the candidates lying closest to c_i.

A discrete random variable X_i is associated with each S_i. The probability that X_i gets the value q_{ik} is denoted by $Pr(q_{ik})$. If some $\mathcal{D}(q_{ik}, c_i) = 0$, in other words the quantizer makes no error for the input color c_i, we set:

$$|S_i| = 1 \text{ and } Pr(q_{i1}) = 1. \tag{3}$$

If $\mathcal{D}(q_{ik}, c_i) \neq 0$, the probability of q_{ik} is inversely proportional to the quantization error that is made if q_{ik} is used as representative:

$$Pr(q_{ik}) \propto \frac{1}{\mathcal{D}(q_{ik}, c_i)}. \tag{4}$$

The exact probability of q_{ik} to become as the representative is:

$$Pr(q_{ik}) = \frac{1}{\mathcal{D}(q_{ik}, c_i) \sum_j \frac{1}{\mathcal{D}(q_{ij}, c_i)}}, \text{ where } j = 1, \cdots, n. \tag{5}$$

4.2 Correction Heuristic

There are situations where the probability assignings made above result in an unpleasant effect. Consider that we are running the 2-best algorithm and for some c_i there are two candidates q_{i1} and q_{i2} with the same distance $\mathcal{D}(q_{i1}, c_i) = \mathcal{D}(q_{i2}, c_i)$. If the distance is very small, human eye cannot notice the change in the color caused by the quantization error. However, the granularity will be disturbing in an area with a solid color because both of the candidates have the same probability to become chosen as the representative. We have experimentally found out that in this situation a simple *correction heuristic* produces better result.

If there are candidates $q_{ik}, q_{il} \in S_i$ such that $\mathcal{D}(q_{ik}, c_i) = \mathcal{D}(q_{il}, c_i)$, we add *correction unit*[2] Δ to the distance of the other candidate:

$$\text{If } \mathcal{D}(q_{ik}, c_i) = \mathcal{D}(q_{il}, c_i) \text{ then } \mathcal{D}(q_{il}, c_i) := \mathcal{D}(q_{il}, c_i) + \Delta, \text{ where } l > k. \tag{6}$$

This procedure is executed as long as there are two candidates with the same distance. If two candidates with same distance are close to the input color, the effect of the correction is considerable. However, if the two candidates lie far from an input color, the effect of the correction is insignificant.

4.3 Projection Heuristic

A straight-forward implementation of the n-best algorithm is not practical because of the large searching space. When the candidates are searched for an input color, $|A| * |B|$ comparisons have to be done. We will present a fast heuristic to approximate the selection process. Another efficient way to do this is described by Thomas [8].

We reduce the search space by considering only the candidates which lie close to an input color. In our heuristic called *projection heuristic*, the search space is reduced from three dimensions to one dimension by projecting the candidates to the rgb axes. For each axis an array of 256 elements is created. When the candidate is projected, the index of the candidate is stored to the array. For example, if we have candidate #24 with rgb color values (47, 56, 244), we store #24 in elements 47, 56, and 244 of the arrays of r, g, and b respectively. If more than one candidates are projected to the same element, the additional ones are stored in a linked list. Similar data structure is often used in open hashing.

[2] Δ is defined on the metrics that is used, it is the smallest possible distance that is greater than zero.

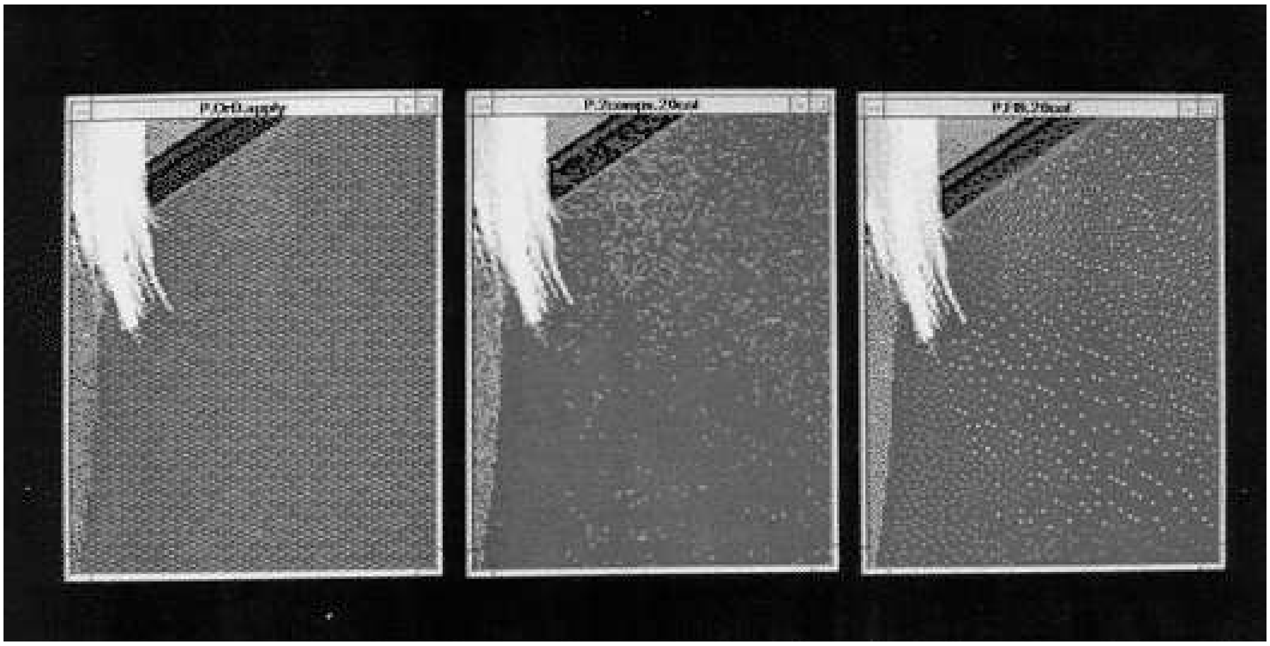

Figure 3: *The patterns generated by the ordered dither of URT, the 2-best dither and the FS algorithm of URT. The original color images are transformed into gray scale images.*

When we want to find the closest candidate for input color c, we project c to the axes and start searching in the neighborhood of c in the arrays of r, g, and b. We call the projection of c in the array r the *initial location* of c in r. Searching is done up and down simultaneously along each axis. When a candidate is found on each of the three axis, we know that it lies near to c in the color space.

In the end of the search more candidates have usually been found than are needed. The found candidates have to be ordered by distance. To make this efficiently we use Manhattan (city-block) distance. This ensures that all the values are integers and only two adding operations for counting a distance are needed. Because the values are integers ordering can be done in linear time with bucket sort.

Also for efficiency we maintain information of the searching process. For every candidate two counters are needed, one for the distance and one for the "hits". When a candidate is seen in one of the arrays the hit counter is incremented and the distance counter is increased by the distance that has been proceeded from the initial location of that array.

When at least n candidates are found and other candidates in B are at least as far in the color space as the nearest of the found candidates, the searching process can be stopped. This procedure ensures that the best approximation for the input color is always present.

5 Experimental results

We compared the n-best algorithm with the *Utah Raster Toolkit*, or *URT* [9, 12], implementations of two previous methods: the ordered dither algorithm (Figure 1) and the FS algorithm. For a subjective comparison we showed to a testing group images that were processed by these chosen algorithms. We also measured processing times of the algorithms.

Some patterns[3] produced by the algorithms are presented in Figure 3. The ordered dither

[3]The original color images can be found for the present at: http://www.cs.Helsinki.FI/~klemstro/patterns.html.

Colors	Method	Face	2*Face	4*Face	8*Face	16*Face
256	F_{URT}^{16}	1	2	6	14	24
	Median cut	3	4	6	15	28
	2-best	7	8	10	21	36
	5-best	8	8	10	22	37
	FS	63	125	257	1009	1990
20	Median cut	3	4	6	15	28
	2-best	5	6	8	20	35
	5-best	6	7	9	21	39
	FS	7	15	34	122	225

Figure 4: *The processing times of the algorithms for test images. All the images have 5 966 colors and the first image has a size of 134 136 pixels. The other images are just multiples of the first one. Observe that the times of the n-best algorithm include also quantizing times.*

produces clearly noticeable symmetric patterns. The FS algorithm produces somehow regular artifacts while the n-best algorithm produces irregular granularity. The amount of the granularity produced by the n-best algorithm can be adjusted with parameter n. The greater n is the more granularity is present. Observe that in the average the n-best algorithm preserves the intensity of the processed image.

Based on our experience, the quality of the final image is usually better for the FS algorithm than for the other two algorithms. Usually the choice $n = 2$ is the best for the n-best algorithm. If the contouring effects are strong, a higher value may be chosen. The quality of the 2-best algorithm is better than ordered dither while the granularity produced by the 5-best algorithm is disturbing according to the testing group[4].

While the quality of the FS algorithm is better than 2-best, the 2-best algorithm is faster. In Figure 4 the processing times of the tested algorithms are given. Running times were measured on an *IBM PowerPC RS6000* with a *RISC*-processor. The bigger the image, the greater the difference between the n-best and the FS algorithms. Ordered dither is very fast, even faster than median cut algorithm.

6 Conclusions and future work

We proposed a new dithering technique for color images. The n-best algorithm adds to the image noise that is proportional to the quantization error. An advantage of the n-best algorithm is that it does not produce any regular patterns to the image.

Experimental results showed that the best value for n is usually 2. When the value of the variable n increases, the contouring effects reduce while the granularity becomes more noticeable. The 2-best algorithm is a good compromise between the speed and the quality of the image. Especially when processing large images, the ratio of quality to speed is good.

The local behavior of the n-best algorithm is a great advantage for applications in which successive images are being processed and the pixels locally change colors, e.g., in full-motion picture, where the correlations of successive frames are high. This is not possible with the FS

[4]For the present some color examples of images produced by different algorithms can be found at: http://www.cs.Helsinki.FI/~klemstro/examples.html.

algorithm because the pixels are not dithered independently.

Because small quantization errors are less disturbing than the granularity produced by the n-best algorithm, a threshold value could be used. If the quantization error is smaller than the threshold, no noise is added to the image. To find advantageous value for the threshold is one of the topics of our future study.

Acknowledgments

We thank Professor Tapani Sarjakoski for bringing up the idea of this work and for his continuous support. We thank Petri Kontkanen for his remarks on Section 4.

References

[1] B. E. Bayer: *An optimum method for two-level rendition of continuous-tone pictures*. In: Proc. IEEE 1973 International Conference on Communications, 1 (1973), 2611–2615.

[2] R. W. Floyd and L. Steinberg: *An adaptive algorithm for spatial gray scale*. In: SID 75 Int. Symp. Dig. Tech. Papers (36) 1975.

[3] M. Gervautz and W. Purgathofer: *A simple method for color quantization: octree quantization*. In: Graphics Gems, (ed.) A. S. Glassner, Academic Press Professional, 1990, 287–293.

[4] P. Heckbert: *Color image quantization for frame buffer display*. In: SIGGRAPH '82 Proceedings, 297–305.

[5] J. F. Jarvis, C. N. Judice, and W. H. Ninke: *A survey of techniques for the display of continuous-tone pictures on bilevel displays*. Computer Graphics and Image Processing, 5 (1976), 13–40.

[6] D. Knuth: *Digital halftones by dot diffusion*. ACM Transaction on Graphics, 6 (1987), 245–273.

[7] V. Ostromoukhov, R. D. Hersch, and I. Amidror: *Rotated dispersed dither: a new technique for digital halftoning*. In: SIGGRAPH '94 Proceedings, 123–130.

[8] S. W. Thomas: *Efficient inverse color map computation*. In: Graphics Gems II, (ed.) J. Arvo, Academic Press, 1991, 116–125.

[9] S. W. Thomas and R. G. Bogart: *Color dithering*. In: Graphics Gems II, (ed.) J. Arvo, Academic Press, 1991, 72–77.

[10] R. A. Ulichney: *Digital halftoning with blue noise*. In: Proceedings of the IEEE, 76 (1988), 56–79.

[11] R. A. Ulichney: *Digital Halftoning*. MIT Press, Cambridge, MA, 1987.

[12] Utah Raster Toolkit, version 3.1. Available via ftp: cs.utah.edu:pub/urt-3.1b.tar.Z.

[13] L. Velho and J. Miranda Gomes: *Digital halftoning with space filling curves*. Computer Graphics, 25, 4 (1991), 81–90.

[14] X. Wu: *Efficient statistical computations for optimal color quantization*. In: Graphics Gems II, (ed.) J. Arvo, Academic Press, 1991, 126–133.