ELSEVIER

Image and Vision Computing 18 (2000) 493–500

image AND vision COMPUTING

www.elsevier.com/locate/imavis

N-Candidate methods for location invariant dithering of color images

K. Lemström[a,1,*], P. Fränti[b,2]

[a]*Department of Computer Science, University of Helsinki, FIN-00014, Helsinki, Finland*
[b]*Department of Computer Science, University of Joensuu, FIN-80101, Joensuu, Finland*

Received 14 December 1998; received in revised form 27 August 1999; accepted 17 September 1999

Abstract

We introduce a new class of dithering methods called *N*-candidate methods. The main idea is that the output color is randomly chosen among several candidate colors so that the estimated color average would be preserved. The dithering process is pixelwise without any interaction with the neighboring pixels. The *N*-candidate methods are thus location invariant, which has two benefits: (1) the algorithm can be fully parallelized; and (2) the image can be partially processed without effecting the pixels outside the processed part. The proposed approach allows more efficient dithering than error diffusion but at the cost of a slightly lower image quality. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Color image quantization; Dithering; *N*-candidate methods; Location invariant; Parallelization

1. Introduction

The classical problem of *color image quantization* aims at representing a *true color* (24 bit) image using a limited number of colors. Quantization is widely used in *non-true color displays* and in *color printers* that cannot reproduce 16 million different colors. *Color palette images* are also faster to operate with, and they require less memory. For example, an image with 256 colors takes 8 bits per pixel, which is only one third of the memory required by a 24-bit true color image of the same size and resolution. Color palette images are also used in computer generated graphics, which require only a limited number of colors.

Color image quantization consists of two steps: *color palette generation* and *color mapping*. The first step aims at finding the optimal set of colors for a given input image (or images). In the second step, the input pixels are mapped to the palette colors. The two steps are relatively independent from each other.

A trivial solution for the color palette generation is to use a predefined color palette which is generated off-line. The image quality, however, is compromised too much if the same palette is applied for all images. It is therefore better to optimize the palette for each input image separately. The color palette optimization is related to the problem of finding *codebook* for a *vector quantizer*, which is well studied in literature [1]. Therefore, most algorithms proposed for vector quantization also apply to the color palette generation.

The second phase, color mapping, can be performed by mapping each color to its nearest palette color. Unfortunately, pixelwise quantization generates visible errors to the image in the form of false contours. The smaller the color palette is the stronger the effect is. Another drawback is that non-palette colors are poorly represented especially if the palette is not well optimized for the input image.

A better solution for color mapping is the use of dithering, in which the image quality is optimized spatially rather than pixelwise. This is sensible because the human eye does not pay so much attention to individual pixels but integrates the colors of several adjacent pixels. Dithering algorithm reduces the visual errors by re-ordering the colors so that their visual combination matches the original image more closely. Dithering has three specific aims:

- to preserve the local color averages;
- to illustrate non-palette colors by the combination of palette colors;
- to prevent false contours caused by quantization.

Dithering methods can be divided into three classes: (1) *dot-pattern dithering* [2,3]; (2) *error diffusion* [4,5]; and (3) *ordered dithering* [6–8]. The methods in the first class emulate the *halftoning process* used in the printing process by representing an input color by a combination of output

* Corresponding author. Tel.: +358-919144209; fax: +358-919144441.

[1] Supported by the Helsinki Graduate School of Computer Science and Engineering, and the Academy of Finland (grant 8745).

[2] Supported by a grant from the Academy of Finland.

PII: S0262-8856(99)00039-6

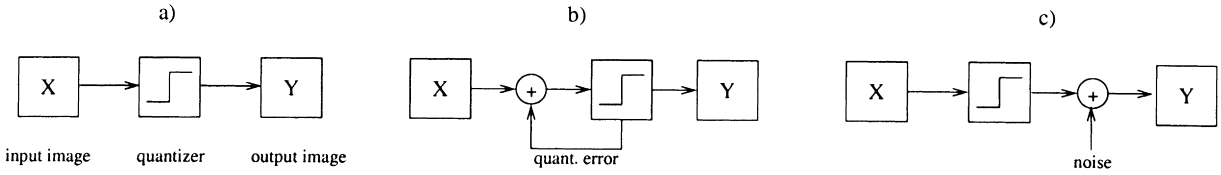

Fig. 1. Block diagrams of (a) quantization; (b) error diffusion dithering; and (c) *N*-candidate dithering.

colors. Error diffusion performs pixelwise quantization by mapping the input pixel to its nearest palette color and then compensating the quantization error to the unprocessed adjacent pixels. Ordered dithering applies pseudo-random thresholding in the selection of the output color.

We introduce a new class of dithering methods called *N-candidate methods*. The dithering is performed by including noise in the quantization process, or in the output of the quantization, see Fig. 1. The new class encloses several existing dithering algorithms [6,9,10]. Ordered dithering [6] is also a special case of the *N-candidate methods*. See Fig. 2 for the classification of the dithering methods.

The idea of *N*-candidate methods is that several candidate colors are selected for each input color. The output color is randomly chosen among the candidates using some weighting criterion. The process is pixelwise without any interaction with the neighboring pixels. The method is location invariant and, unlike error diffusion, fully supports parallel implementation. Another advantage is that, due to randomization, the method does not produce any regular patterns in the output image, which is typical for the ordered dithering.

The rest of the paper is organized as follows. We start in Section 2 by defining basic concepts of the problem and by giving a brief summary of the existing color image quantization methods. The general framework of the new *N*-candidate methods is introduced in Section 3 followed by a detailed discussion of the various parameters of the method. We also discuss a new adaptive *N*-candidate algorithm, which optimizes the number of candidates for each input color separately. In Section 4, we make objective (numerical) and subjective (visual) comparisons between the *N*-candidate methods and the other methods. Conclusions are then drawn in Section 5.

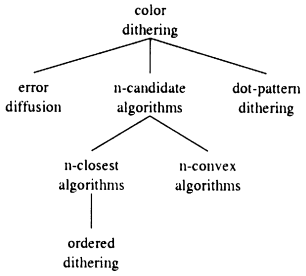

Fig. 2. Classification tree of dithering methods.

2. Color image quantization

We consider digitized *RGB images* consisting of red (R), green (G), and blue (B) components. The color components are typically represented by 8 bits each, giving about 16.6 million different colors. The following symbols will be used throughout this paper:

x_i: pixel of the *input image* $X = \{x_i | i = 1, 2, \ldots, M\}$,
c_j: color in the *color palette* $C = \{c_j | j = 1, 2, \ldots, K\}$,
f: mapping from an input pixel x_i to output color c_j : $f(x) : X \mapsto C$.

It is commonly assumed that the pixels are located in the *Euclidean space*. The Euclidean distance d between pixels x_k and x_l can be calculated as:

$$d(x_k, x_l) = \sqrt{(x_k[\mathrm{r}] - x_l[\mathrm{r}])^2 + (x_k[\mathrm{g}] - x_l[\mathrm{g}])^2 + (x_k[\mathrm{b}] - x_l[\mathrm{b}])^2}, \quad (1)$$

where r, g, and b refer to the red, green and blue color components, respectively.

2.1. Problem setup

Color image quantization can be formally defined as an optimization problem as follows. Given an input image X, and a palette of size K, find a color palette C and a mapping function f, maximizing the visual quality of the output image. Unfortunately, there is no exact mathematical formulation for the 'visual quality' in this context. The three specific aims given in the introduction can be used as intuitive goals.

Usually the color palette and the mapping function are optimized separately since they are considered independent processes. The color palette is generated without any prior

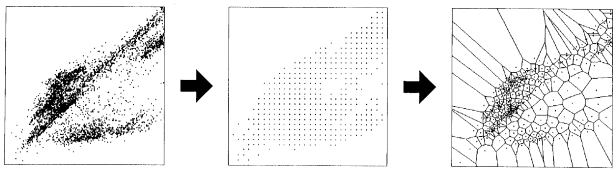

Fig. 3. Illustration of the color palette generation. The leftmost picture represents RG-plotting of the colors of the input image. In the middle the colors are prequantized to 5 bits per color component. The resulting color palette is plotted in the rightmost picture. The Voronoi diagram illustrates the boundaries of the color clusters.

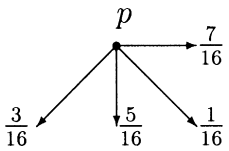

Fig. 4. Weighting of the Floyd–Steinberg error diffusion.

knowledge of the dithering applied. This is illustrated in Fig. 3. Once the palette is created, the color mapping tries to optimize the quality of the output image using the given color palette. However, there are no principle objections to why the knowledge of the dithering algorithm could not be taken into account when designing the palette. For one such attempt, see Ref. [11].

An advantage of separate optimizations is that the color palette optimization can be mathematically formulated. The goal is to find a color palette C which minimizes the pixel-wise quantization errors. An optimal mapping for a single pixel can then be defined as:

$$f_{\text{opt}}(x_i) = \{c_j | d(x_i, c_j) \leq d(x_i, c_h)\}, \qquad \forall h = 1, 2, \ldots, K. \quad (2)$$

Furthermore, we assume that the aim is to minimize the squared *average* distance between the input colors and their nearest palette color. An exact formulation of the color palette optimization can then be defined as the minimization of *mean square error* (MSE):

$$C = \left\{ f(x_i) | \min\left(\frac{1}{M} \sum_i (d(x_i, f(x_i))^2) \right) \right\}. \quad (3)$$

2.2. Color palette generation

Splitting methods [12–15] begin by initializing the palette with only one color, which is the centroid (weighted average) of all input colors. The palette is then iteratively enlarged by a splitting procedure until it reaches the size K. For example, the *median-cut algorithm* [14] splits the color space into rectangular subregions aiming at cells with an approximately equal number of pixels. Gervautz and Purgathofer have proposed a straightforward algorithm that uses the *octree* data structure which is commonly used in image processing [13]. The algorithm by Wu and Zhang [15], on the other hand, does not restrict the direction of the splitting at all, but it performs *principal component analysis* for finding the direction of maximal variance.

An opposite, bottom–up approach, starts by initializing a palette where each input color is represented by its own color. The palette is then iteratively reduced by merging two neighboring colors at a time. The process is repeated until the palette reaches the size K. For example, *pairwise nearest neighbor* (PNN) [16] uses local optimization and combines the colors that increase the distortion least. The method is simple to implement and yields high quality color palettes, but at the cost of high running time.

Probably the most widely used method is the *generalized Lloyd algorithm* [17], also referred to as GLA, or LBG *algorithm*. It begins with an initial color palette, which is iteratively improved by following two optimization steps. In the *partition step*, the color space regions are updated by remapping each input color to its nearest palette color. In the *centroid step*, the color palette is replaced by the centroids of the new regions obtained in the partition step. These two steps are repeated until no further improvement is achieved.

The initial palette for GLA can be generated by any existing method. A set of randomly chosen input colors is commonly used. *Heckbert's quantization algorithm* [14], on the other hand, uses the median-cut algorithm for producing the initial palette, which is then iterated by the GLA. The use of various splitting techniques has been studied in Ref. [12].

2.3. Color mapping

Dot-pattern dithering: In dot-pattern dithering the image is divided into square blocks (e.g. 2×2) and each block is processed separately. The average color of the block is calculated and the pixels are jointly mapped to a combination of palette colors so that the local color average inside the block would be preserved [2,3].

Error diffusion: Error diffusion algorithms map each input pixel to its nearest palette color. The quantization error is compensated by diffusing its opposite to the unprocessed adjacent pixels. For instance, *Floyd–Steinberg* algorithm [4] processes the image in scan-raster order. The quantization error is divided into four fractions and their opposites are added to the neighboring pixels to the current pixel, see Fig. 4.

The main drawback of error diffusion is that it is not location invariant and therefore it does not support parallel processing. The problem can be partially solved by dividing the image into blocks and processing them separately. Knuth's *dot diffusion* [5] processes the pixels in a predefined order so that the error is propagated to two pixels far from the block boundaries.

Ordered dithering: In ordered dithering the image is divided into square $n \times n$ blocks of fixed size, typically 4×4 or 8×8. The blocks are quantized separately using a predefined *threshold matrix*. The ordered dither matrix can be considered as a pseudo-random noise generator. The thresholding is designed for gray-scale images, but the idea can be generalized to color images as well, as shown by Orchard and Bouman [6]. The thresholding is performed by finding the two nearest colors in the color palette, and applying a pseudo-random thresholding between these two colors.

3. *N*-Candidate methods

The *N*-candidate methods process the image pixelwise in any order. Each input pixel is quantized separately without any interaction to the quantization of other pixels. For each input pixel, several (N) candidate colors are selected from

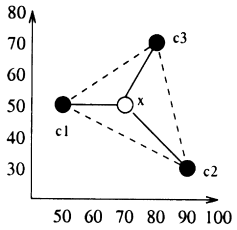

Fig. 5. Illustration of 3-candidate dithering. The input pixel x is reproduced using the colors c_1, c_2 and c_3.

the color palette. The output color is chosen among the candidates by a randomized process that aims at minimizing the error between the input pixel and the *estimated average* of the output color. The quantization error of a single pixel is not vital as we aim at preserving the local color averages.

The N-candidate methods are location invariant since the quantization of an input pixel x_i is independent of the spatial location of the pixel, and the quantization result of all other pixels x_j. There are two main advantages of the locality. Firstly, the process can be fully parallelized. Secondly, one can process an arbitrary part of the image without effecting the pixels outside the processed part.

The algorithmic structure of the N-candidate methods is as follows:

alg 1 (*N*-**candidate**)
1. **for** *each* x_i **do begin**
2. *Select N candidate colors* $R_i = \{r_i^k | 1 \le k \le N\}$ *from C*
3. *Assign weight* w_i^k *for each candidate* r_i^k *so that* $\sum_k w_i^k = 1$
4. *Select random candidate from* R_i *using the weights* w_i^k
5. **end**

The candidates r_i^k should be chosen so that the input color x_i could be reproduced by a linear combination of the candidate colors in a convenient ratio, see Fig. 5. The equality cannot be guaranteed in general because we use a randomized process. Instead, we can expect that the estimated color average equals the input color if the candidates are properly chosen. As a side effect, the process generates unavoidable random noise, which is inherent in any dithering process.

The N-candidate methods consist of three subproblems that must be solved: (1) how many candidate colors are needed; (2) how are they chosen; and (3) how are they weighted in the randomizing. These questions will be considered in the following subsections.

3.1. Selecting the candidate colors

The way the candidates are chosen depends on the chosen optimization criteria. The following properties should be considered: (1) local color averages should be preserved; (2) quantization noise should be minimized; (3) edge structures should be retained; and (4) false contours should be prevented. The first property is the main motivation of most dithering methods. The second criterion, however, clearly contradicts the main idea of the N-candidate algorithms. Nevertheless, it should also be kept in mind when designing the algorithm. The last two properties are also important, but since the method operates pixelwise, there is no direct way to model spatial structures of the image. Instead, these two goals are met indirectly and the optimization concentrates merely on the first two properties.

A straightforward approach minimizes the distance between the input pixel and the candidate colors by selecting the N closest colors from the color palette [9,18]. We call this approach the *N-closest method*. A more sophisticated approach is to select the candidate colors from different directions around the input color. The aim is to establish a convex hull from the candidate colors that surround the input color. In this way, the input color can be reproduced as a linear combination of the candidates. Variants of this method are denoted here as the *N-convex methods*. The tetrahedron algorithm by Purgathofer and Gröller [10] is an example of the N-convex methods (see also Ref. [19]).

We propose next a greedy algorithm for selecting the candidates for the N-convex methods. The candidates are chosen one at a time by taking the closest palette color to x_i as the first candidate. The next color is chosen so that the centroid of the selected candidates is located as close to x_i as possible. The following algorithm describes the search task. The variable z records the optimal position for the next candidate to be chosen.

alg 2 (**GreedyCandidateSelection** (x_i))
1. $z := x_i$
2. $R_i := \varnothing$

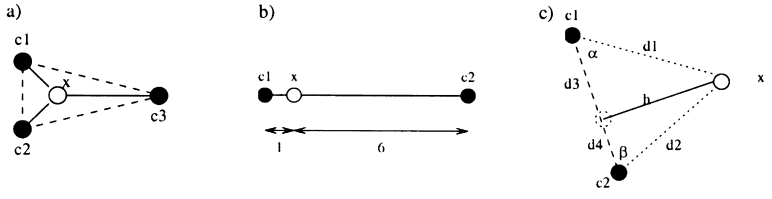

Fig. 6. Different situations when selecting the candidate colors: (a) The first two candidates are located close to the original color but the third color is also needed for reproducing the input color; (b) Only one candidate should be chosen since the second candidate is too far from the first color; (c) This example illustrates how the two candidates should be weighted so that the distance from x to their centriod would be minimized.

3. **for** $k := 1$ **to** N **do begin**
4. *Find* r_i^k *as the unused palette color closest to* z
5. *Mark* r_i^k *as used*
6. $R_i := R_i \cup r_i^k$
7. $z := z + (x_i - r_i^k)$
8. **end**
9. **return** R_i

The greedy algorithm forms a candidate set by a joint minimization of the *N*-closest and *N*-convex criteria. The average color preservation is the primary goal and it is achieved if the candidate set forms an *N*-convex whose geometric center in the RGB cube matches the input color. The input color can be reproduced only if it lies inside the convex hull formed by the candidates. The quantization noise is minimized as the secondary criterion by always taking the closest color meeting the *N*-convex criterion.

3.2. Number of candidates

The question of a proper number of candidate colors has two aspects. More candidates mean higher quantization errors but, on the other hand, a very few candidates may not be sufficient to reproduce the input color. Consider the 2D example in Fig. 6a. The two closest colors c_1 and c_2 are very close to the input color x but their color average matches the input color only in the vertical direction. The third candidate c_3 is therefore needed to reproduce the input color as the linear combination of the candidate colors. In general, $N + 1$ candidates are sufficient to represent any color in an *N*-dimensional Euclidean space if the candidates are properly chosen.

A high number of candidates is theoretically well argued by the *N*-convex criterion but it still has some serious drawbacks in practice. It is sometimes not possible to find three or more candidates that are close enough to the input color, see Fig. 6b. If some candidates are chosen far from the input color, they will rarely be selected in the randomizing. In this case, the color average matches only in a relatively large area and smaller parts may be reproduced by a color that has little to do with the original color. This can also cause severe quantization noise. The number of colors should therefore be limited to very few.

Orchard and Bouman have considered ordered dithering for color images in [6]. If this method is applied to an arbitrary color palette, it can be seen as a 2-closest method. Lemström et al. [9] have generalized this to any number of *N* by introducing the *N*-closest method, which they refer to as the *N*-best method. Their experiments have shown that two candidates are usually the best choice for maximizing the visual quality. The tetrahedron algorithm by Purgathofer and Gröller [10] generates a 3D *dual Voronoi diagram* for a given palette. The method connects colors closest to each other by forming tetrahedrons in the RGB space. The input colors inside a tetrahedron can be represented by a linear combination of the vertices of the given tetrahedron. This method is therefore a 4-convex method. The extreme case $N = 1$ should be applied in areas where the color is constant and well represented by a single palette color.

The above discussion implies that the number of candidates should be chosen adaptively. Given a maximum number of candidates $N_{\max}$, and an upper limit for the allowed quantization error $e_{\max}$, an adaptive variant of the *N*-convex method can be constructed as follows. The distance from the input color to the centroid of the candidate set is measured and recorded for all $N = 1, \ldots, N_{\max}$. The set minimizing the distance is chosen. In practice, we use the greedy selection algorithm and select the candidate colors one at a time. Any candidate whose distance to x_i exceeds $e_{\max}$ is discarded, and no more candidates will be considered. The upper limit is set relative to the quantization error of the closest color. From experiments we have found $e_{\max} = 5 * e_1$ a good choice.

3.3. Randomizing between the colors

The output color is randomly chosen among the candidates using a weighting function. The weight w_i determines the probability for the candidate r_i to be chosen as the output color in the randomizing process. Following from the *N*-convex criterion the weights should be assigned so that the expected color average equals to the input color. If the input color lies inside the convex hull of the candidate colors, a unique set of weights can be solved using the methods of linear algebra. In other cases, the problem is reduced to finding the nearest point in the border of the convex hull, which leads to the optimization problem of finding a set of weights $w_i^k \in [0, 1]$ so that it would minimize:

$$\min\{d(x_i, \sum(r_i^k * w^k)) \mid \sum(w^k) = 1\}. \quad (4)$$

The result of formula (4) can be rather complex in general. The exact weighting, however, is not so vital for the visual quality and therefore we prefer a weighting function that can be easily computed. We propose the following approximation, in which the weight w_i^k for a candidate c_i^k is inversely proportional to its distance d_i^k to the input color:

$$w_k = \frac{d_k^{-1}}{\sum(d_i^{-1})} \quad (5)$$

This proposed weighting function is a feasible solution for minimizing both the *N*-convex criterion and the quantization noise. The further the chosen color is from the input color the smaller is its weight.

The randomizing itself is both an advantage and disadvantage. In comparison to the deterministic processes of ordered dithering and error diffusion, the randomizing does not generate any visible block boundaries or undesired background texture. For smaller palettes, on the other hand, random appearance may sometimes be visually disturbing around sharp edges. Randomized dithering can destroy edge structures that would be easier to recognize if the dithering resulted in regular background patterns.

Table 1
The test images and their statistics

Image	Size	Pixels	Unique colors
Colored balls	568 × 564	320352	2028
Yacht	480 × 512	245760	150053
Parrots	768 × 512	98304	50187

4. Experiments

In the following experiments we use the three test images: Colored balls, Yacht and Parrots. The first image serves as a special case with smooth color transitions. The last two images, on the other hand, are rich in colors; more than 50% of the pixels have a unique color, see Table 1. The image Yacht contains a large number of small details and high contrast edges. The image Parrots is smoother but it has high color saturation. For each image we generate two color palettes using the standard GLA method [17]. The number of colors are 16 and 256 (see Fig. 3 for an illustration of a 2D plotting of the 256 color palette for Parrots).

The following methods are included in the comparisons:

1. 2-closest (2-clo);
2. 2-convex (2-con);
3. *N*-convex (*N*-con);
4. Floyd–Steinberg (FLS).

The *2-closest* method uses two candidates and always selects the two closest colors. Its alternative is the *2-convex* method, which selects the candidates primarily in order to preserve the local color averages. The *N*-convex refers to the method that selects the number of candidates adaptively using the *N*-convex criterion. All methods exclude candidates whose distance e_i to the input color exceeds the limit $e_{\max} = 5 * e_1$. FLS refers to the standard Floyd–Steinberg dithering and it is a natural point of comparison.

Numeric comparison: The behavior of the *N*-candidate methods is first studied by comparing how often they select a different candidate than the closest one. Statistics for the image Yacht are shown in Table 2. The first candidate is chosen about 65–75% of the time when using 256 color palette, and about 82–85% of the time when using 16 color palette. There is not much variation between the methods. In general, the 2nd color in the *N*-convex methods is further than that of the 2-closest method and it is therefore less frequently chosen because of the weighting scheme. In the *N*-convex method, the third candidate is chosen seldom, and the 4th and 5th very seldom.

Table 2 gives comparative results for the FLS method also. In principle, the FLS always selects the closest color but because of the error diffusion, this is not the case in practice. The FLS chooses the first candidate more often than the *N*-candidate methods do. The FLS, however, is blind to the ordering of the colors and therefore chooses other candidates also.

A numeric comparison of the distortion caused by the methods is summarized in Table 3. The quantization error measures how much noise the dithering adds to the image. The results show that the *N*-candidate methods generate slightly more noise than the FLS on an average. However, in certain special cases the situation can be the opposite; e.g.

Table 3
Comparison of pixelwise and blockwise errors of the different methods. Quantization error refers to the mean square error between the input and output pixels, whereas the average error corresponds to the mean squared error between the color averages in every 3 × 3 block of the image. The values are given as PSNR (peak signal-to-noise ratio)

	Colored balls	Yacht	Parrots	Colored balls	Yacht	Parrots
	Quantization errors (PSNR), 16 colors			Quantization errors (PSNR), 256 colors		
FLS	22.457	21.775	20.949	29.919	30.526	39.655
2-clo	22.820	21.534	20.926	29.554	30.245	39.200
2-con	22.722	21.454	20.855	29.318	30.091	39.011
N-con	22.615	21.126	20.615	28.630	29.531	38.592
	Average errors (PSNR), 16 colors			Average errors (PSNR), 256 colors		
FLS	28.455	26.312	24.862	47.949	36.604	35.919
2-clo	27.299	25.565	24.044	47.758	35.744	35.181
2-con	27.343	25.789	24.378	47.676	35.188	35.658
N-con	27.402	25.727	24.398	47.441	35.866	35.478

Table 2
The running numbers of the selected candidates for different methods

Yacht, 16 colors

#	2-clo	2-con	*N*-con	FLS
1	205897	207934	202122	215209
	83.78%	84.61%	82.24%	87.57%
2	39863	37826	35187	26742
	16.22%	15.39%	14.32%	10.88%
3	–	–	6058	3452
			2.47%	1.40%
4	–	–	1844	308
			0.75%	0.13%
5	–	–	549	46
			0.22%	0.02%

Yacht, 256 colors

#	2-clo	2-con	*N*-con	FLS
1	177035	185763	161824	201437
	72.04%	75.59%	65.85%	81.96%
2	68725	59997	49350	33084
	27.96%	24.41%	20.08%	13.46%
3	–	–	21643	8266
			8.81%	3.36%
4	–	–	9270	2186
			3.77%	0.89%
5	–	–	3673	519
			1.49%	0.21%

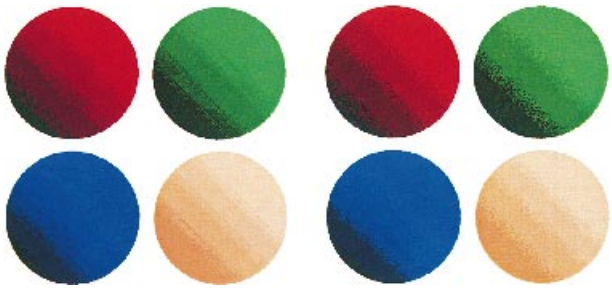

Fig. 7. Colored balls. To the left is the result of the error diffusion, and to the right *N*-convex algorithm (16 colors).

Colored balls with 16 colors, see Fig. 7. The size of the color palette, on the other hand, has by far much greater impact on the quality than the choice between the dithering methods.

The preservation of the color averages is probably more important than the pixelwise differences. The differences between the FLS and the *N*-convex method is about 0.8 dB in favor to FLS, see Table 3. The slight advantage of the FLS method originates from the fact that it compensates the quantization error immediately to the local neighborhood. The *N*-candidate methods, on the other hand, are location invariant and therefore it cannot utilize the actual quantization result of the previous pixels. The 2-convex method is slightly better than the 2-closest method but the difference is rather small. The use of more candidates is not supported by the experiments as the *N*-convex method is not capable of improving over the 2-convex method.

Visual comparison: Overall, the error diffusion and the *N*-convex methods are of the same visual quality in the case of 256 color palette, see Fig. 8. The images can be differentiated by the type of the dithering noise. The *N*-convex methods generate noise with random appearance whereas the distortion generated by the FLS has more regular patterns. This becomes more observable when the size of the color palette is reduced to 16 colors. The images dithered by the 2-convex and the 2-closest methods (not shown here) are of the same visual quality and do not differ from that of the *N*-convex method.

Fig. 8. At the top from left to right: Parrots 16 colors (a) FLS; (b) *N*-convex. Illustration of the error type produce by FLS and *N*-convex method. The detailed images, from left right: (a) FLS, $K = 256$; (b) *N*-convex; $K = 256$; (c) FLS, $K = 16$; and (d) *N*-convex, $K = 16$.

Although the dithering texture of the FLS method is disturbing, it is in some case also useful. In the case of 16 colors, the edges in the image can be recognized easier in the image dithered by the FLS because of the regularity of the dithering noise. The same edge structures are harder to detect in the images dithered by the *N*-candidate methods because of their randomness, see Fig. 9a and b. This problem could be solved by using a location dependent pseudo-random generator as experimented in Fig. 9c and d. Using this modification, the algorithm would preserve its parallel nature but it would not be location invariant anymore. It is noted that the problem of the distorted edge structures is not visible with 256 colors and the modification is therefore not necessary in this case.

The upper limit for the quantization error was found to be important in the *N*-candidate methods. For some input pixels there are only one candidate that is close enough. If the other candidates are not excluded they will occasionally be selected and cause disturbing artifacts in the image in the form of isolated and random noise pixels.

5. Conclusions

A new class of dithering methods is introduced. The methods are denoted as *N*-candidate methods since they use several candidate colors for each input pixel. The randomizing ensures that the method does not produce any regular patterns in the output image. The *N*-candidate methods are location invariant and fully support parallel implementation. The proposed approach therefore allows more efficient dithering than error diffusion but at the cost of a slightly lower image quality. The difference is mostly unnoticeable in the case of 256 colors but can be disturbing when the number of available colors is limited. The method is therefore not recommended for printing purposes.

Fig. 9. At the top from left to right: Yachts 16 colors (a) FLS; (b) 2-convex. In the detailed images the effect of the randomizing versus pseudo-randomizing in the case of 16 colors are illustrated. From left to right: (a) the FLS; (b) the 2-convex method with randomizing; (c) the 2-convex method with a 1D pseudo-random generator; (d) the 2-convex method with a 2D pseudo-random generator.

On the basis of our experiments we conclude that the choice of the parameters is not crucial for the performance of the algorithm. There are no noticeable differences in quality between the images produced by the *N*-closest and the *N*-convex criteria. The number of candidates can also be limited to two without any side-effects. However, it was found to be important to set a upper limit for the quantization error, and in this way also exclude the second candidate when its distance is much larger than that of the first candidate. Otherwise the dithering would produce disturbing high frequency noise.

References

[1] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer, Dordrecht, 1992.
[2] L. Velho, J. Gomes, Stochastic screening dithering with adaptive clustering, SIGGRAPH'95, 1995, pp. 273–276.
[3] G. Wyvill, C. McNaughton, Three plus five makes eight: a simplified approach to halftoning, in: N.M. Patrikalakis (Ed.), Scientific Visualization of Physical Phenomena, Springer, Berlin, 1991, pp. 379–392.
[4] R.W. Floyd, L. Steinberg, An adaptive algorithm for spatial gray scale, SID'75 Int. Symp. Dig. Tech. Papers, 1975.
[5] D. Knuth, Digital halftones by dot diffusion, ACM Transaction on Graphics 6 (1987) 245–273.
[6] M.T. Orchard, C.A. Bouman, Color quantization of images, IEEE Transactions on Signal Processing 39 (1991) 2677–2690.
[7] V. Ostromoukhov, R.D. Hersch, I. Amidror, Rotated dispersed dither: a new technique for digital halftoning, SIGGRAPH'94, 1994, pp. 123–130.
[8] W. Purgathofer, R. Tobler, M. Geiler, Improved threshold matrices for ordered dithering, in: A. Paeth (Ed.), Graphic Gems, V, Academic Press, New York, 1995, pp. 297–301.
[9] K. Lemström, J. Tarhio, T. Takala, Color dithering with *n*-best algorithm, Proceedings of Fourth International Conference in Central Europe on Computer Graphics and Visualization, Plzen, Czech Republic, 1996, pp. 162–170.
[10] W. Purgathofer, E. Gröller, Using tetrahedrons for dithering color pictures, Automatika 29 (1-2) (1988) 45–50.
[11] L. Akarun, D. Ozdemir, O. Yalcin, Modified quantisation algorithm for dithering of colour images, Electronics Letters 32 (13) (1996) 1185–1186.
[12] P. Fränti, T. Kaukoranta, O. Nevalainen, On the splitting method for vector quantization codebook generation, Optical Engineering 36 (11) (1997) 3043–3051.
[13] M. Gervautz, W. Purgathofer, A simple method for color quantization: octree quantization, in: A.S. Glassner (Ed.), Graphics Gems, Academic Press, New York, 1990, pp. 287–293.
[14] P. Heckbert, Color image quantization for frame buffer display, SIGGRAPH'82, 1982, pp. 297–305.
[15] X. Wu, K. Zhang, A better tree-structured vector quantizer, IEEE Proceedings of Data Compression Conference, 1991, pp. 392–401.
[16] W.H. Equitz, A new vector quantization clustering algorithm, IEEE Transactions on Acoustics, Speech, and Signal Processing 37 (10) (1989) 1568–1575.
[17] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Transactions on Communications 28 (1) (1980) 84–95.
[18] N. Goldberg, Colour image quantization for high resolution graphics display, Image and Vision Computing 9 (5) (1991) 303–312.
[19] H. Kang, Color Technology for Electronic Imaging Devices, SPIE Press, 1996.