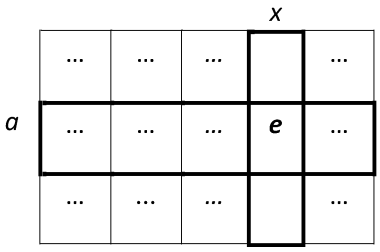
IEEE Photonics Journal
An IEEE Photonics Society Publication

Lossless Compression of Dithered Images

Volume 5, Number 3, June 2013

Basar Koc
Ziya Arnavut
Huseyin Kocak



DOI: 10.1109/JPHOT.2013.2264276
1943-0655/$31.00 ©2013 IEEE

Lossless Compression of Dithered Images

Basar Koc,[1] Ziya Arnavut,[2] and Huseyin Kocak[1]

[1]Department of Computer Science, University of Miami, Miami, FL 33146 USA
[2]Department of Computer Science, SUNY Fredonia, Fredonia, NY 14063, USA

DOI: 10.1109/JPHOT.2013.2264276
1943-0655/$31.00 ©2013 IEEE

Manuscript received March 29, 2013; revised May 6, 2013; accepted May 7, 2013. Date of publication May 20, 2013; date of current version June 25, 2013. Corresponding author: B. Koc (e-mail: b.koc@umiami.edu).

Abstract: In order to display high-bit resolution images on low-bit resolution displays, bit resolution needs to be reduced. This problem is vital especially for low-cost or small (mobile) devices. To untangle the bit reduction problem, special color quantization algorithms, called dithering, are employed on high-bit resolution images. The dithering process helps to remedy the problem, but it does not help much in terms of storage and transmission of images. To reduce storage needs and lower data transmission, numerous special compression techniques have been proposed in the last several decades. While the well-known compression algorithms, such as gzip, help lower image file sizes, usually, they are not adequate. To improve the compression gain, special compression techniques that take into account structure of image data must be developed. In this paper, we show that, when the pseudo-distance technique (PDT) is used for dithered images, it yields better compression results than GIF and PNG.

Index Terms: Color palette, lossless image compression, dithering, pseudo-distance metric, GIF, PNG.

1. Introduction

For long term storage and efficient transmission of data, data compression is critical. Over the past decade, several compression techniques have been proposed. Some of the researchers focused on general data compression, while others designed more specific file compression techniques for movies, sound, and image files. When a file type is known, better compression results can be attained with less effort because better prediction models can be designed.

Color image quantization [1], [2] is an essential technique for low-cost display devices. A low-cost display device has limited ability. For example, there are 16.8 million (2^{24}) possible color combinations in an RGB color image. Since an RGB color image consists of three bytes for each pixel; one byte for each color (red, blue, and green), a color image quantization is applied to map 24 bits to 16, 8, or 4 bits per pixel.

A color image quantization is a three-step process: the first step is designing a suitable palette to reduce the number of colors, the second step is coding, and the final step is decoding. For each RGB image, by generating a set of representative colors, a palette is formed. Later, the palette is used in image encoding and decoding. In the encoding process, each color pixel in the RGB image is replaced with the index of the closest color in the palette. In effect, each RGB color pixel is represented by an index, which is mapped to the color palette. If the palette size is 256 or less, each index can be represented with a byte. Hence, effectively, image size is reduced from 3 bytes per color to one byte per index value by applying the quantization. In quantization process, we may

have quantization errors, e.g., distortion. To overcome quantization errors, dithering algorithms are used. In a dithered image, colors not available in the palette are approximated by a diffusion of colored pixels from within the available palette. The human eye perceives the diffusion as a mixture of the colors within it.

In recent years, several dithering algorithms were proposed. The most well-known algorithm [3] was developed in 1976 by Floyd–Steinberg, which is known as the Floyd–Steinberg dithering technique in literature. It uses error-diffusion algorithms instead of basic dithering algorithms [2], such as average, ordered, or random, and produces images which look closer to original form. In [4], for color error diffusion problem, a vector-based approach was proposed by Akarun *et al.* In [5], Alasseur showed that noise shaping method improves the image quality, while in [6], Yao proposed a method which provides optimal solution for the error diffusion problem.

In World Wide Web, many Web pages use quantized images. They are stored and transmitted after they are compressed losslessly with the standard Graphics Interchange Format (GIF) or Portable Network Graphics (PNG). Since GIF was patented, the PNG format was designed to replace GIF as an improved and patent-free alternative. The PNG file for an image is usually 10%–35% smaller than the GIF file for the same image. These algorithms fail to obtain good compression gain, e.g., GIF standard uses Lempel–Ziv compression, which treats the image as a 1-D sequence of index values, ignoring the 2-D nature. Besides GIF and PNG, several researchers proposed different methods for quantized images. According to the survey paper of Pinho and Neves [7], Memon's and Venkateswaran's re-indexing scheme yields the best compression gain when JPEG-LS or JPEG-2000 is used after the re-indexing method. Memon and Venkateswaran [8] treated the problem as a general smoothness maximization problem, while Zeng *et al.* method [9] used re-indexing based on index-differences. All these methods show that reordering and re-indexing help 2-D compression schemes. In [10], Battiato *et al.* show that PNG yields better compression than JPEG-based techniques on re-indexed color-mapped images.

While several different techniques have been explored for color quantized and dithered images, the use of pseudo-distance metric has not been investigated for dithered images since it was proposed by Kuroki *et al.* [11]. In our earlier work [12], [13], we showed that better compression gain than other techniques can be obtained when pseudo-distance technique (PDT) is applied to color-mapped images. In [14], we examined PDT for images with different dithering techniques. By using a dynamic color-table, we obtained better results than the other well-known techniques, such as GIF and PNG. In this paper, we show that further improvement on compression gain is possible when the color-table is updated based on the neighboring pixels.

This paper is planned as follows: In Section 2, the description of the PDT and a prediction algorithm along with a modified PDT for encoding images are presented. In Section 3, decoding process is explained. In Section 4, the coder we employed is described. In Section 5, the presentation of our experimental result is given, and in Section 6, the conclusions of our paper are summarized.

2. Encoding Algorithm

First, we form a distance matrix **D** as shown in Table 1, by calculating Euclidean distance [12] between every pair of indices from a color palette (color-map table). In each row of **D**, there may be similar values. To overcome the problem of non-uniqueness of the entries in rows of **D**, we compute a corresponding pseudo-distance matrix **P**, where each row element $\mathrm{P}(a, b)$ gets a unique value c $(0 \leq c \leq 255)$ based on the rank of $\mathrm{D}(a, b)$ in the sorted row a of matrix **D**, as shown in Table 2.

Currently used prediction-based image compression techniques usually predict the current pixel x by using some neighbors of x, as shown in Fig. 1. In [11], Kuroki *et al.* used the pixel a (reference) and x to determine a prediction error e from the pseudo-distance matrix, as shown in Fig. 2.

Instead of using a as the reference pixel for determining x, if we choose the minimum of $\{\mathrm{P}(a, x), \mathrm{P}(c, x)\}$ for each color-index x, compression gain can be improved further. But this decision requires transmission of extra information, which reverses the gains obtained from selecting the minimum of $\{\mathrm{P}(a, x), \mathrm{P}(c, x)\}$. In [12], to avoid any overhead, we employed Gradient Adjusted Predictor (GAP). The goal of a predictor algorithm is to predict x with the minimum

TABLE 1

Example of a distance matrix **D**

	0	**1**	**2**	**3**	**..**	**254**	**255**
0	0	5.81	3.52	6.56	..	200.86	204.51
1	5.81	0	8.99	11.61	..	201.24	205.07
..	..	..	..	..	..	..	..
254	200.86	201.24	198.89	196.47	..	0	7.55
255	204.51	205.07	202.45	199.96	..	7.55	0

TABLE 2

Pseudo-distance matrix **P**

	0	**1**	**2**	**3**	**..**	**254**	**255**
0	0	2	1	3	..	254	255
1	1	0	2	3	..	254	255
..	..	..	..	..	..	..	..
254	254	255	253	252	..	0	1
255	254	255	253	252	..	1	0

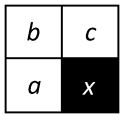


Fig. 1. Predicted pixel x and reference neighboring pixels.

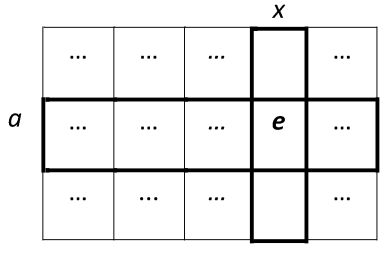


Fig. 2. Ranking with pseudo-distance matrix **P**.

possible error. We experimented with various techniques, including, for instance, the Paeth algorithm [15], to construct a superior prediction scheme. We concluded that the *PREDICT* method below results in the best prediction of x from its neighbors:

PREDICT(a, b, c)
1 **if** $a = b$
2 **then return** c
3 **if** $b = c$
4 **then return** a

```
5  if a = c
6     then return b
7  return c
```

In [14], we used a procedure in PDT with dynamic pseudo-distance matrix. Recently, our experiments showed that a further compression gain over our work in [14] is possible, when the matrix is updated using the *Pseudo-Distance-Transform* method below:

```
Pseudo-Distance-Transform()
 1  repeat
 2        p ← Predict(a, b, c)
 3        e ← P[p, x]
 4       for i ← 1 to n
 5          do if e > P[p, e_i]
 6             then P[p, e_i] ← P[p, e_i] + 1
 7
 8        P[p, x] ← 0
 9       if p ≠ c
10          then e_2 ← P[c, x]
11             for i ← 1 to n
12                do if e_2 > P[c, c_i]
13                   then P[c, c_i] ← P[c, c_i] + 1
14
15             P[c, x] ← 0
16    until All pixels processed
17 return
```

This algorithm, while preserving the uniqueness of entries of the rows, may increase the number of zeros when the same colors occur again as neighbors.

3. Decoding Algorithm

The decoding process reconstructs the original image file as follows: By using the index table, first, we construct the pseudo-distance matrix **P**. Let r be the first value retrieved from the index table. In encoding process, r was encoded as a raw byte. To recover the first row and first column of the image, we first receive the error signal e of the next pixel from the encoded image. Then, in row r of the pseudo-distance matrix, we search for the error signal value e. Upon determining e in row r, we emit the corresponding column value x as the original index value for the image to be reconstructed. Finally, we let x be r. These steps are formalized in the *Reverse-Pseudo-Distance-Transform* method:

```
Reverse-Pseudo-Distance-Transform()
 1  repeat
 2        e ← P[r, e]
 3       for i ← 1 to n
 4          do if e > r_i
 5             then P[r, e_i] ← P[r, e_i] + 1
 6
 7       Emit corresponding column value x as the original index value
 8        x ← r
 9    until First row and first column processed
10 return
```

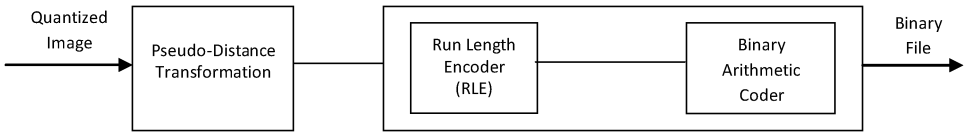


Fig. 3. Steps in encoding of an image.

TABLE 3

Compression results of images dithered with Floyd–Steinberg (normal) method, in bits per pixel (bpp)

Images	# of colors	Size	GIF	PNG	PDT + BAC	Proposed PDT + BAC
Benjerry	48	28,326	1.239	1.077	0.923	**0.858**
Books	7	29,338	3.046	2.956	2.836	**2.826**
Clegg	256	719,158	5.766	4.919	4.009	**3.938**
Cwheel	256	481,078	2.907	2.404	2.187	**2.116**
Descent	122	64,542	2.928	2.639	2.363	**2.255**
Fractal	256	389,190	6.923	5.882	5.093	**5.056**
Frymire	256	1,238,678	2.708	2.191	1.940	**1.852**
Gate	84	61,302	2.939	2.581	2.251	**2.201**
Ghouse	256	481,078	5.181	4.531	4.061	**4.000**
Music	8	6,302	2.482	2.126	2.227	**2.054**
Netscape	32	61,382	2.113	1.948	1.837	**1.759**
Party8	12	75,606	0.854	0.675	0.590	**0.546**
Pc	6	1,721,858	1.694	1.004	1.002	**0.874**
Sea_dusk	46	157,538	0.323	0.081	0.028	**0.025**
Serrano	256	502,886	2.970	2.520	2.171	**2.053**
Sunset	204	308,070	2.601	2.125	1.839	**1.764**
Winaw	10	148,894	0.997	0.980	0.889	**0.833**
Yahoo	229	28,110	1.983	1.740	1.468	**1.392**
Average		361,296	2.759	2.354	2.095	**2.022**
W. Avg.			3.115	2.516	2.215	**2.127**
Normalized			1.465	1.183	1.041	**1.000**

After this process, we attain first row and first column values of the original image. To decode the rest of the rows, the following procedure is applied:

REVERSE-PSEUDO-DISTANCE-TRANSFORM()

```
1  repeat
2       r ← PREDICT(neighborA_x, neighborB_x, neighborC_x)
3       e ← P[r, e]
4      for i ← 1 to n
5        do if e > r_i
6          then P[r, e_i] ← P[r, e_i] + 1
7
8       P[r, e] ← 0
9      if r ≠ c
10       then e_2 ← P[c, e]
11         for i ← 1 to n
12           do if e_2 > P[c, c_i]
```

TABLE 4

Compression results of images dithered with Floyd–Steinberg (reduced color bleeding) method, in bits per pixel (bpp)

Images	# of colors	Size	GIF	PNG	PDT + BAC	Proposed PDT + BAC
Benjerry	48	28,326	1.239	1.077	0.923	**0.858**
Books	7	29,338	3.046	2.956	2.836	**2.826**
Clegg	256	719,158	5.751	4.905	3.993	**3.921**
Cwheel	256	481,078	2.889	2.380	2.164	**2.095**
Descent	122	64,542	2.928	2.639	2.363	**2.255**
Fractal	256	389,190	6.923	5.882	5.093	**5.056**
Frymire	256	1,238,678	2.688	2.176	1.925	**1.837**
Gate	84	61,302	2.939	2.581	2.251	**2.201**
Ghouse	256	481,078	5.188	4.529	4.058	**4.000**
Music	8	6,302	2.482	2.126	2.227	**2.054**
Netscape	32	61,382	2.113	1.948	1.837	**1.759**
Party8	12	75,606	0.854	0.675	0.590	**0.546**
Pc	6	1,721,858	1.694	1.004	1.002	**0.874**
Sea_dusk	46	157,538	0.323	0.081	0.028	**0.025**
Serrano	256	502,886	2.961	2.509	2.169	**2.050**
Sunset	204	308,070	2.601	2.125	1.839	**1.764**
Winaw	10	148,894	0.997	0.980	0.889	**0.833**
Yahoo	229	28,110	1.983	1.740	1.468	**1.392**
Average		361,296	2.756	2.351	2.092	**2.019**
W. Avg.			3.108	2.509	2.209	**2.121**
Normalized			1.465	1.183	1.041	**1.000**

then $P[c, c_i] \leftarrow P[c, c_i] + 1$

$P[c, e] \leftarrow 0$

Emit corressponding column value x as the original index value

$x \leftarrow r$

until *All pixels processed*

return

Finally, the original color-mapped image is reconstructed by the decoding algorithm without losing any information.

4. Coder

In compression of transformed image data, we used context-based binary arithmetic coder (BAC) with a run-length encoder (RLE). The use of RLE is vital in entropy coder phase since the output of the PDT on dithered images contains a significant number of zeros. RLE detects long run sequences and regroup the repetitive sequences. It also improves the context-based arithmetic coder (AC) performance by helping the AC in determining non-zero symbol probabilities. While the use of RLE improves compression gain, it also decreases the computation time of the entropy coder. Fenwick proposed a special run-length encoding algorithm on 0 s and 1 s in [16]. In [17], the zero-run-length transformed data were used along with the context-based AC. In this paper, we used a binary AC (BAC) with run-length encoding algorithm similar to the one in [17]. The steps involved compressing an image are depicted schematically in Fig. 3.

TABLE 5

Compression results of images dithered with positioned dithering method, in bits per pixel (bpp)

Images	# of colors	Size	GIF	PNG	PDT + BAC	Proposed PDT + BAC
Benjerry	48	28,326	1.239	1.077	0.923	**0.858**
Books	7	29,338	3.046	2.956	2.836	**2.826**
Clegg	256	719,158	5.569	4.482	3.779	**3.689**
Cwheel	256	481,078	2.392	1.553	1.844	**1.820**
Descent	122	64,542	2.928	2.639	2.363	**2.255**
Fractal	256	389,190	6.923	5.882	5.093	**5.056**
Frymire	256	1,238,678	2.142	1.426	1.498	**1.454**
Gate	84	61,302	2.939	2.581	2.251	**2.201**
Ghouse	256	481,078	4.536	3.748	3.616	**3.588**
Music	8	6,302	2.482	2.126	2.227	**2.054**
Netscape	32	61,382	2.113	1.948	1.837	**1.759**
Party8	12	75,606	0.854	0.675	0.590	**0.546**
Pc	6	1,721,858	1.694	1.004	1.002	**0.874**
Sea_dusk	46	157,538	0.323	0.081	0.028	**0.025**
Serrano	256	502,886	2.307	1.614	1.667	**1.600**
Sunset	204	308,070	2.601	2.125	1.839	**1.764**
Winaw	10	148,894	0.997	0.980	0.889	**0.833**
Yahoo	229	28,110	1.983	1.740	1.468	**1.392**
Average		361,296	2.615	2.146	1.986	**1.922**
W. Avg.			2.848	2.131	2.008	**1.936**
Normalized			1.471	1.101	1.037	**1.000**

5. Experimental Results

In this paper, we evaluated the performance of our algorithm on a set of commonly used test images. These images were obtained from ftp://ftp.ieeta.pt/~ap/images/. They were converted from the portable pixel map (PPM) format to color-index-based bitmap (BMP) format using *Gimp* (ver. 2.6.11). In conversion process, the Floyd–Steinberg, the Floyd–Steinberg reduced color, and the positioned dithering techniques were used.

The Paeth prediction algorithm is used in PNG, while GAP is used in CALIC, and Median is used in JPEG-LS to determine which color index is used in prediction of x. In PDT transformation phase, we experimented with these prediction algorithms. However, when we applied the *PREDICT* algorithm, we improved compression gain than the other predictors.

After the PDT transformation, the zero value is dominant. Since we obtained more zeros with the transformation, we applied a context-modeled BAC with an RLE, as in [17], on the PDT transformed dithered images to take advantage of zeros. Using this process, we achieved better compression gain than the well-known image compression schemes, such as GIF, PNG, JPEG-LS, and JPEG2000. In Tables 3–5, we present the results of our experiments.

6. Conclusion

The PDT is an efficient lossless image compression method for color-palette images since the PDT runs in linear time and requires only one-pass [11]. In our previous work [14], we have shown that, when the PDT is used in conjunction with a context-model BAC, we obtained better compression results than the well-known image compressors such as GIF, PNG, JPEG-LS, and JPEG2000 on dithered images. In this paper, we have shown that further compression gains are possible when we update one more neighbor of the predicted pixel in PDT matrix. Indeed, with this modification, on

three sets of dithered images, we have achieved on average 3.9% improvement over our previous work resulting in total gains of 46.8% over GIF and 14.2% over PNG.

References

[1] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Trans. Signal Process.*, vol. 39, no. 12, pp. 2677–2690, Dec. 1991.
[2] [Online]. Available: http://www.visgraf.impa.br/Courses/ip00/proj/Dithering1/
[3] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial grey scale," in *Proc. Soc. Inf. Display*, 1976, vol. 17, pp. 75–77.
[4] L. Akarun, Y. Yardunci, and A. E. Cetin, "Adaptive methods for dithering color images," *IEEE Trans. Image Process.*, vol. 6, no. 7, pp. 950–955, Jul. 1997.
[5] C. Alasseur, A. G. Constantinides, and L. Husson, "Colour quantisation through dithering techniques," in *Proc. ICIP*, Sep. 14–17, 2003, vol. 1, pp. I-469–I-472.
[6] Z. Yao and Y. Wan, "A high performance dithering method for gray and color image quantization," in *Proc. 6th Int. Conf. WiCOM*, Sep. 23–25, 2010, pp. 1–4.
[7] A. J. Pinho and A. J. R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1411–1418, Nov. 2004.
[8] N. D. Memon and A. Venkateswaran, "On ordering color maps for lossless predictive coding," *IEEE Trans. Image Process.*, vol. 5, no. 11, pp. 1522–1527, Nov. 1996.
[9] W. Zeng, J. Li, and S. Lei, "An efficient color re-indexing scheme for palette-based compression," in *Proc. Int. Conf. Image Process.*, 2000, vol. 3, pp. 476–479.
[10] S. Battiato, G. Gallo, G. Impoco, and F. Stanco, "An efficient re-indexing algorithm for color-mapped images," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1419–1423, Nov. 2004.
[11] N. Kuroki, T. Yamane, and M. Numa, "Lossless coding of color quantized images based on pseudo distance," in *Proc. 47th MMWSCAS*, Jul. 25–28, 2004, vol. 1, pp. I-245–I-247.
[12] B. Koc and Z. Arnavut, "Gradient adjusted predictor with pseudo-distance technique for lossless compression of color-mapped images," in *Proc. FIT*, Dec. 19–21, 2011, pp. 275–280.
[13] B. Koc and Z. Arnavut, "A new context-model for the pseudo-distance technique in lossless compression of color-mapped images," in *Proc. SPIE Opt. Eng.+* Appl., Appl. Digit. Image Process. XXXV, Aug. 12–16, 2012, p. 84 991O.
[14] B. Koc, Z. Arnavut, and H. Kocak, "Lossless compression of dithered images with the pseudo-distance technique," in *Proc. 9th Int. Conf. HONET*, Dec. 12–14, 2012, pp. 147–151.
[15] A. W. Paeth, "Image file compression made easy," in *Graphics GEMS II*. New York, NY, USA: Academic, 1991.
[16] P. Fenwick, "The Burrows–Wheeler transform for block sorting text compression—principles and improvements," *Comput. J.*, vol. 39, no. 9, pp. 731–740, 1996.
[17] S. Deorowicz, "Second step algorithms in the Burrows–Wheeler compression algorithm," *Softw. Practice Exper.*, vol. 32, no. 2, pp. 99–111, Feb. 2001.