ELSEVIER

Information Processing Letters 81 (2002) 13–21

Information Processing Letters

www.elsevier.com/locate/ipl

Fast 2D discrete cosine transform on compressed image in restricted quadtree and shading format

Kuo-Liang Chung [a,*], Wen-Ming Yan [b,1]

[a] *Department of Information Management, Institute of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672*

[b] *Department of Computer Science and Information Engineering, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei, Taiwan 106*

Received 15 January 2001; received in revised form 6 March 2001
Communicated by A.A. Bertossi

Abstract

Given a compressed image in the restricted quadtree and shading format, this paper presents a fast algorithm for computing 2D discrete cosine transform (DCT) on the compressed grey image directly without the need to decompress the compressed image. The proposed new DCT algorithm takes $O(K^2 \log K + N^2)$ time where the decompressed image is of size $N \times N$ and K denotes the number of nodes in the restricted quadtree. Since commonly $K < N$, the proposed algorithm is faster than the indirect method by decompressing the compressed image first, then applying the conventional DCT algorithm on the decompressed image. The indirect method takes $O(N^2 \log N)$ time. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Algorithms; Discrete cosine transform; Gouraud shading; Quadtree; Compressed image

1. Introduction

Given a binary image of size $N \times N$ $(= 2^n \times 2^n)$, based on the quadtree spatial data structure [10], the binary image can be represented in a compressed form. Many efficient algorithms [11] have been developed on the compressed binary image in quadtree format directly without the need to decompress it. However, in the literature, only few algorithms on the compressed grey image domain in the quadtree format were presented. In [2,9], the $N \times N$ grey image is considered. According the restricted quadtree format, which is a complete quadtree, the grey image is partitioned into $2^s \times 2^s$ $(= S \times S)$ squares, $s \leqslant n$, where each square is of size $2^{n-s} \times 2^{n-s}$ and each pixel within the square has the same grey level. Using the subsampling technique, the subsampled image with size $2^s \times 2^s$ is called the compressed image in the restricted quadtree format. Anguh [2] and Philips [9] presented two different algorithms for Fourier transform, although the proposed two algorithms have the same time complexity, $O(S^2 \log S + N^2)$.

[*] Corresponding author. Supported by NSC89-2213-E011-062.
E-mail addresses: klchung@cs.ntust.edu.tw (K.-L. Chung), ganboon@csie.ntu.edu.tw (W.-M. Yan).
[1] Supported by NSC87-2119-M002-006.

The experimental results [3] reveal that combining the quadtree structure and Gouraud shading technique [5] has better compression effect. That is, given the same grey image of size $N \times N$, we have $K \leqslant S$ when comparing the compressed image with size $K \times K$ in the restricted quadtree and shading format and the compressed image with size $S \times S$ in the restricted quadtree format. In addition, the decompressed image in [3] can avoid the blocking effect, i.e., has better image quality. In this research, we focus on the 2D discrete cosine transform (DCT) [1, 12,7,6,13,4,8,14] and the compressed image domain in the restricted quadtree and shading format which is more general than the previous compressed domain [2,9]. The DCT on the original $N \times N$ image can be performed in $\mathrm{O}(N^2 \log N)$ time. The motivation of this research is to design an efficient 2D DCT algorithm on the compressed image in the restricted quadtree and shading format directly without the need to decompress it.

On the compressed image in the restricted quadtree and shading format, this paper presents a novel and fast 2D DCT algorithm and the proposed algorithm takes $\mathrm{O}(K^2 \log K + N^2)$ time. Since commonly $K < N$, the proposed DCT algorithm is faster than the indirect method by decompressing the compressed image first, then applying the conventional DCT algorithm [1] on the decompressed image. The indirect method takes $\mathrm{O}(N^2 \log N)$ time. The main contribution of this paper is that the compression effect in the compressed image implies that the proposed DCT algorithm has a computation-saving effect when compared to the indirect method. However, the compressed image in the restricted quadtree and shading format considered in this paper is a special case of the one in quadtree and shading format [3]. In [3], the constructed quadtree may be incomplete and it is still an open problem to design an efficient algorithm for DCT transform on such a compressed image.

2. The compressed image in restricted quadtree and shading format

Following the shading concept used in [3], the given image is first augmented by duplicating the original last column and the original last row to become of size $(N+1) \times (N+1)$. Then the augmented image is partitioned into $2^k \times 2^k = K \times K$ overlapping homogeneous squares, i.e., subimages, as few as possible, where each square is of size $(2^{n-k}+1) \times (2^{n-k}+1) = (M+1) \times (M+1)$. Here, $N = K \times M$. For each square, we only save the grey levels of its four corners. Each corner's grey level of one square is shared by the neighboring three corners in the neighboring squares. Thus, any homogeneous square shares its top edge with the northern homogeneous square's bottom edge, and so on. The formal definition of a homogeneous square [3] will be given in next paragraph. Using Gouraud shading method [5], the grey levels of the pixels within one square can be interpolated using the the four corners' grey levels of that square. When compared to the compressed image in the restricted quadtree format [2, 9], the purpose of making the blocks overlap can alleviate the blockiness effect [3] when employing the shading technique and leads to better image quality. Throughout this paper, the subsampled image in the restricted quadtree and shading format is called the subsampled image without confusion.

It is known that each subimage has four corners, namely, the left-bottom corner, the right-bottom corner, the left-upper corner, and the right-upper corner. The grey levels of the pixels within one square can be interpolated using the the four corners' grey levels at positions (x_1, y_1), (x_2, y_1), (x_1, y_2), and (x_2, y_2); their grey levels are g_1, g_2, g_3, and g_4, respectively. Here, $x_2 = x_1 + M$ and $y_2 = y_1 + M$. Let

$$g_5 = g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1) \quad \text{and} \quad g_6 = g_3 + \frac{g_4 - g_3}{x_2 - x_1}(x - x_1),$$

thus the estimated grey level of the pixel at (x, y) within the subimage is calculated as

$$f(x, y) = g_5 + \frac{g_6 - g_5}{y_2 - y_1}(y - y_1). \tag{1}$$

Given a specified error tolerance ε, if $|g(x, y) - f(x, y)| \leqslant \varepsilon$ holds for all the estimated pixels at position (x, y) in the subimage, $x_1 \leqslant x < x_2$ and $y_1 \leqslant y < y_2$, then the subimage is homogeneous. Here, $g(x, y)$ denotes the original grey level at position (x, y). From (1), we have

$$f(x, y) = a_0 + a_1(x - x_1) + a_2(y - y_1) + a_3(x - x_1)(y - y_1), \tag{2}$$

where

$$a_0 = g_1, \qquad a_1 = \frac{g_2 - g_1}{x_2 - x_1}, \qquad a_2 = \frac{g_3 - g_1}{y_2 - y_1}, \qquad a_3 = \frac{g_4 - g_3 - g_2 + g_1}{(x_2 - x_1)(y_2 - y_1)}.$$

Using the above restricted quadtree and shading approach to compress the grey image with size $N \times N$, we obtain the compressed image with size $K \times K$. Among these $K \times K$ squares, for each square, only four corners' grey levels are required to be stored. Totally there are $4K^2$ grey levels to be stored and these $4K^2$ grey levels will be used as the input of our proposed DCT algorithm. Following the same assumption as in [2,9], we omit the preprocessing time for preparing the subsampled image.

3. The proposed algorithm

For exposition, we let $0 \leqslant k_x, k_y < 2^k$, $x_1 = k_x M$, $x_2 = k_x M + M$, $y_1 = k_y M$, and $y_2 = k_y M + M$. According to the shading method mentioned in (1), the grey level $f(x_1 + l_x, y_1 + l_y)$, $0 \leqslant l_x, l_y < M$, at position $(x_1 + l_x, y_1 + l_y)$ is interpolated using the four corners' grey levels, $g_1 = g(x_1, y_1)$, $g_2 = g(x_2, y_1)$, $g_3 = g(x_1, y_2)$, and $g_4 = g(x_2, y_2)$. Since $x_2 - x_1 = y_2 - y_1 = M$, by (2), if we let

$$\begin{aligned} a_0(k_x, k_y) &= g_1 = g(x_1, y_1), \qquad a_1(k_x, k_y) = \frac{g_2 - g_1}{x_2 - x_1} = \frac{g(x_2, y_1) - g(x_1, y_1)}{M}, \\ a_2(k_x, k_y) &= \frac{g_3 - g_1}{y_2 - y_1} = \frac{g(x_1, y_2) - g(x_1, y_1)}{M}, \\ a_3(k_x, k_y) &= \frac{g_4 - g_3 - g_2 + g_1}{(y_2 - y_1)(x_2 - x_1)} = \frac{g_4 - g_3 - g_2 + g_1}{M^2}, \end{aligned} \tag{3}$$

then we have

$$\begin{aligned} f(k_x M + l_x, k_y M + l_y) &= f(x_1 + l_x, y_1 + l_y) \\ &= a_0(k_x, k_y) + a_1(k_x, k_y) l_x + a_2(k_x, k_y) l_y + a_3(k_x, k_y) l_x l_y \end{aligned} \tag{4}$$

for $0 \leqslant l_x, l_y < M$. Thus, the approximate image with grey levels $f(x, y)$'s is used to help the computation of the 2D DCT. From (3) and (4), $f(k_x M + l_x, k_y M + l_y)$ can be written as

$$\begin{aligned} f(k_x M + l_x, k_y M + l_y) &= c_0(k_x, k_y) + c_1(k_x, k_y)\left(l_x - \frac{M-1}{2}\right) \\ &\quad + c_2(k_x, k_y)\left(l_y - \frac{M-1}{2}\right) + c_3(k_x, k_y)\left(l_x - \frac{M-1}{2}\right)\left(l_y - \frac{M-1}{2}\right), \end{aligned}$$

where

$$\begin{aligned} &c_0(k_x, k_y) = a_0(k_x, k_y) + \bigl(a_1(k_x, k_y) + a_2(k_x, k_y)\bigr)\frac{M-1}{2} + a_3(k_x, k_y)\left(\frac{M-1}{2}\right)^2, \\ &c_1(k_x, k_y) = a_1(k_x, k_y) + a_3(k_x, k_y)\frac{M-1}{2}, \qquad c_2(k_x, k_y) = a_2(k_x, k_y) + a_3(k_x, k_y)\frac{M-1}{2}, \\ &c_3(k_x, k_y) = a_3(k_x, k_y). \end{aligned} \tag{5}$$

The concerning $N \times N$ 2D DCT is defined by $F(u, v) = \frac{2}{N} e(u) e(v) \overline{F}(u, v)$, where

$$e(k) = \begin{cases} \dfrac{1}{\sqrt{2}} & k = 0, \\ 1 & 1 \leqslant k \leqslant N-1, \end{cases}$$

and

$$\overline{F}(u,v)=\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1}\sum_{l_x=0}^{M-1}\sum_{l_y=0}^{M-1} f(k_xM+l_x,k_yM+l_y)\cos\frac{\pi(2k_xM+2l_x+1)u}{2N}\cos\frac{\pi(2k_yM+2l_y+1)v}{2N}.$$

From the trigonometric formula, we have

$$\cos\frac{\pi(2k_xM+2l_x+1)u}{2N} = \cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2l_x+1-M)u}{2N}-\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2l_x+1-M)u}{2N}.$$

Similarly, it is easy to derive

$$\cos\frac{\pi(2k_yM+2l_y+1)v}{2N} = \cos\frac{\pi(2k_y+1)v}{2K}\cos\frac{\pi(2l_y+1-M)v}{2N}-\sin\frac{\pi(2k_y+1)v}{2K}\sin\frac{\pi(2l_y+1-M)v}{2N}.$$

Let

$$h_1(k_x,k_y)=\sum_{l_x=0}^{M-1}\sum_{l_y=0}^{M-1} f(k_xM+l_x,k_yM+l_y)\cos\frac{\pi(2l_x+1-M)u}{2N}\cos\frac{\pi(2l_y+1-M)v}{2N},$$

$$h_2(k_x,k_y)=\sum_{l_x=0}^{M-1}\sum_{l_y=0}^{M-1} f(k_xM+l_x,k_yM+l_y)\cos\frac{\pi(2l_x+1-M)u}{2N}\sin\frac{\pi(2l_y+1-M)v}{2N},$$

$$h_3(k_x,k_y)=\sum_{l_x=0}^{M-1}\sum_{l_y=0}^{M-1} f(k_xM+l_x,k_yM+l_y)\sin\frac{\pi(2l_x+1-M)u}{2N}\cos\frac{\pi(2l_y+1-M)v}{2N},$$

$$h_4(k_x,k_y)=\sum_{l_x=0}^{M-1}\sum_{l_y=0}^{M-1} f(k_xM+l_x,k_yM+l_y)\sin\frac{\pi(2l_x+1-M)u}{2N}\sin\frac{\pi(2l_y+1-M)v}{2N},$$

then we have

$$\begin{aligned}\overline{F}(u,v) ={}& \sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_1(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K}\\ &-\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_2(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K}\\ &-\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_3(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K}\\ &+\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_4(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K}.\end{aligned}$$

From (5), it is known that

$$f(k_xM+l_x,k_yM+l_y) = c_0(k_x,k_y)+c_1(k_x,k_y)\left(l_x-\frac{M-1}{2}\right) + c_2(k_x,k_y)\left(l_y-\frac{M-1}{2}\right)+c_3(k_x,k_y)\left(l_x-\frac{M-1}{2}\right)\left(l_y-\frac{M-1}{2}\right),$$

so we have

$$h_1(k_x,k_y)=c_0(k_x,k_y)S_0(u,v)+c_1(k_x,k_y)S_1(u,v)+c_2(k_x,k_y)S_2(u,v)+c_3(k_x,k_y)S_3(u,v),$$

where

$$\begin{aligned}
S_0(u,v) &= \sum_{l_x=0}^{M-1}\cos\frac{\pi(2l_x+1-M)u}{2N}\sum_{l_y=0}^{M-1}\cos\frac{\pi(2l_y+1-M)v}{2N},\\
S_1(u,v) &= \sum_{l_x=0}^{M-1}\left(l_x-\frac{M-1}{2}\right)\cos\frac{\pi(2l_x+1-M)u}{2N}\sum_{l_y=0}^{M-1}\cos\frac{\pi(2l_y+1-M)v}{2N},\\
S_2(u,v) &= \sum_{l_x=0}^{M-1}\cos\frac{\pi(2l_x+1-M)u}{2N}\sum_{l_y=0}^{M-1}\left(l_y-\frac{M-1}{2}\right)\cos\frac{\pi(2l_y+1-M)v}{2N},\\
S_3(u,v) &= \sum_{l_x=0}^{M-1}\left(l_x-\frac{M-1}{2}\right)\cos\frac{\pi(2l_x+1-M)u}{2N}\sum_{l_y=0}^{M-1}\left(l_y-\frac{M-1}{2}\right)\cos\frac{\pi(2l_y+1-M)v}{2N}.
\end{aligned}$$

In what follows, we want to show that the closed forms of $S_1(u,v)$, $S_2(u,v)$, and $S_3(u,v)$ are zeros and want to derive the condensed forms of $h_1(k_x,k_y)$, $h_2(k_x,k_y)$, $h_3(_x,k_y)$, and $h_4(k_x,k_y)$. Let

$$A_c(t)=\sum_{x=0}^{M-1}\cos\frac{\pi(2x+1-M)t}{2N} \quad \text{and} \quad B_c(t)=\sum_{x=0}^{M-1}\left(x-\frac{M-1}{2}\right)\cos\frac{\pi(2x+1-M)t}{2N}.$$

Lemma 1. *$A_c(t)=\sin(\pi Mt/2N)/\sin(\pi t/2N)$ and $B_c(t)=0$.*

Proof. See Appendix A. □

Plugging the two closed forms in Lemma 1 into $S_0(u,v)$, $S_1(u,v)$, $S_2(u,v)$, and $S_3(u,v)$, it yields to $S_0(u,v)=A_c(u)A_c(v)$, $S_1(u,v)=B_c(u)A_c(v)=0$, $S_2(u,v)=A_c(u)B_c(v)=0$, and $S_3(u,v)=B_c(u)B_c(v)=0$. Thus, we have $h_1(k_x,k_y)=c_0(k_x,k_y)A_c(u)A_c(v)$. By the same arguments, let

$$A_s(t)=\sum_{x=0}^{M-1}\sin\frac{\pi(2x+1-M)t}{2N} \quad \text{and} \quad B_s(t)=\sum_{x=0}^{M-1}\left(x-\frac{M-1}{2}\right)\sin\frac{\pi(2x+1-M)t}{2N},$$

we have the following result.

Lemma 2. *$A_s(t)=0$ and the closed form of $B_s(t)$ is equal to*

$$-\frac{N}{\pi}\frac{\frac{\pi M}{2N}\sin\frac{\pi t}{2N}\cos\frac{\pi Mt}{2N}-\frac{\pi}{2N}\cos\frac{\pi t}{2N}\sin\frac{\pi Mt}{2N}}{\sin^2\frac{\pi t}{2N}}.$$

Proof. See Appendix B. □

Therefore, we have $h_2(k_x,k_y)=c_2(k_x,k_y)A_c(u)B_s(v)$, $h_3(k_x,k_y)=c_1(k_x,k_y)B_s(u)A_c(v)$, and $h_4(k_x,k_y)=c_3(k_x,k_y)B_s(u)B_s(v)$.

From the above derivations, we have

$$\begin{aligned}&\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_1(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)u}{2K}\\&\quad=A_c(u)A_c(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_0(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K},\end{aligned}$$

$$\begin{aligned}&\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_2(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)u}{2K}\\&\quad=A_c(u)B_s(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_2(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K},\end{aligned}$$

$$\begin{aligned}&\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_3(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)u}{2K}\\&\quad=B_s(u)A_c(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_1(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K},\end{aligned}$$

$$\begin{aligned}&\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} h_4(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)u}{2K}\\&\quad=B_s(u)B_s(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_3(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K}.\end{aligned}$$

Consequently, we have

$$\begin{aligned}\overline{F}(u,v)\;=\;&A_c(u)A_c(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_0(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K}\\&-A_c(u)B_s(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_2(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K}\\&-B_s(u)A_c(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_1(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K}\\&+B_s(u)B_s(v)\sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_3(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K}.\end{aligned}$$

For simplifying notations used in $\overline{F}(u,v)$, let

$$C_0(u,v) = \sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_0(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K},$$

$$C_1(u,v) = \sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_1(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\cos\frac{\pi(2k_y+1)v}{2K},$$

$$C_2(u,v) = \sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_2(k_x,k_y)\cos\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K},$$

$$C_3(u,v) = \sum_{k_x=0}^{K-1}\sum_{k_y=0}^{K-1} c_3(k_x,k_y)\sin\frac{\pi(2k_x+1)u}{2K}\sin\frac{\pi(2k_y+1)v}{2K},$$

the above four equalities concern one $K \times K$ 2D DCT, one $K \times K$ discrete sine and cosine transform (DSCT), one $K \times K$ discrete cosine and sine transform (DCST), and one $K \times K$ discrete sine transform (DST), respectively. All the related $K \times K$ DCT, DSCT, DCST, and DST can be computed in $\mathrm{O}(K^2 \log K)$ time [1]. Originally, for $C_0(u,v)$, we have the range limitation: $0 \leqslant u < K$ and $0 \leqslant v < K$. Because of $C_0(K,v) = C_0(u,K) = 0$, the feasible range of u and v could be extended to $0 \leqslant u, v \leqslant v$. For $C_1(u,v)$, we have the range limitation: $0 < u \leqslant K$ and $0 \leqslant v < K$. From $C_1(0,v) = C_1(u,K) = 0$, the feasible range of u and v could be extended to $0 \leqslant u, v \leqslant K$. Similarly, from $C_2(K,v) = C_2(u,0) = 0$, and $C_3(0,v) = C_3(u,0) = 0$, for $C_2(u,v)$ and $C_3(u,v)$, the feasible range of u and v could be extended to $0 \leqslant u, v \leqslant K$.

However, we have to extend the feasible range of u and v from 0 to $N-1$. It is easy to verify that $C_0(2K-u,v) = -C_0(u,v)$, $C_0(u,2K-v) = -C_0(u,v)$, $C_1(2K-u,v) = C_1(u,v)$, $C_1(u,2K-v) = -C_1(u,v)$, $C_2(2K-u,v) = -C_2(u,v)$, $C_2(u,2K-v) = C_2(u,v)$, $C_3(2K-u,v) = C_3(u,v)$, and $C_3(u,2K-v) = C_3(u,v)$. Using the above eight equalities, the feasible range of u and v can be extended to 0 to $2K-1$. Similarly, it can be verified that $C_i(u+2K,v) = -C_i(u,v)$ and $C_i(u,v+2K) = -C_i(u,v)$ for $i = 0,1,2,3$ and the feasible range of u and v can be extended to 0 to $4K-1$. By the same arguments, from $C_i(u+4K,v) = C_i(u,v)$ and $C_i(u,v+4K) = C_i(u,v)$ for $i = 0,1,2,3$, the feasible range of u and v can be extended to 0 to $N-1$. Then from the simplified forms of $h_1(k_x,k_y)$, $h_2(k_x,k_y)$, $h_3(k_x,k_y)$, and $h_4(k_x,k_y)$, it yields to

$$\overline{F}(u,v) = A_c(u)A_c(v)C_0(u,v) - A_c(u)B_s(v)C_2(u,v) - B_s(u)A_c(v)C_1(u,v) + B_s(u)B_s(v)C_3(u,v). \tag{6}$$

It takes $\mathrm{O}(N^2)$ time for computing (6). In addition, it also takes $\mathrm{O}(N^2)$ time for computing $F(u,v) = \frac{2}{N}e(u)e(v)$ $\overline{F}(u,v)$ for $0 \leqslant u, v \leqslant N-1$. Consequently, we have the main result.

Theorem 1. *On the compressed image in the restricted quadtree and shading format, the proposed 2D DCT algorithm takes* $\mathrm{O}(K^2 \log K + N^2)$ *time, where the decompressed grey image is of size* $N \times N$ *and* K *denotes the number of nodes in the restricted quadtree.*

The compressed image domain considered in this paper is more general than the previous domain [2,9] since in the previous restricted quadtree format with size $S \times S$, each quadrant is of constant grey value. The experimental results in [3] reveal $K \leqslant S$. This better compression effect implies that the proposed DCT algorithm has a computation-saving effect. Since commonly $K < N$, the proposed DCT algorithm is faster than the indirect method by decompressing the compressed image first, then applying the conventional DCT algorithm on the decompressed image. The indirect method takes $\mathrm{O}(N^2 \log N)$ time. However, in [3], the constructed quadtree may be incomplete, so it is still an open problem to design an efficient DCT algorithm on such a compressed image.

Acknowledgements

The authors appreciate the referees and the Editor Professor Alan Bertossi for their valuable comments that lead to the improved version of the paper.

Appendix A. The proof of Lemma 1

Since

$$A_c(t) = \sum_{x=0}^{M-1} \cos \frac{\pi(2x+1-M)t}{2N},$$

we have

$$\begin{aligned} 2\sin\frac{\pi t}{2N} A_c(t) &= \sum_{x=0}^{M-1} 2\cos\frac{\pi(2x+1-M)t}{2N} \sin\frac{\pi t}{2N} \\ &= \sum_{x=0}^{M-1}\left[\sin\frac{\pi(2x+2-M)t}{2N} - \sin\frac{\pi(2x-M)t}{2N}\right] \\ &= \sin\frac{\pi M t}{2N} - \sin\frac{\pi(-Mt)}{2N} = 2\sin\frac{\pi M t}{2N}. \end{aligned}$$

Therefore, the closed form of $A_c(t)$ is given by

$$A_c(t) = \frac{\sin\frac{\pi M t}{2N}}{\sin\frac{\pi t}{2N}}.$$

From

$$B_c(t) = \sum_{x=0}^{M-1}\left(x - \frac{M-1}{2}\right)\cos\frac{\pi(2x+1-M)t}{2N},$$

let $y = M-1-x$, then we have

$$\begin{aligned} B_c(t) &= \sum_{y=0}^{M-1}\left(M-1-y-\frac{M-1}{2}\right)\cos\frac{\pi(2(M-1-y)+1-M)t}{2N} \\ &= \sum_{y=0}^{M-1}\left(\frac{M-1}{2}-y\right)\cos\frac{\pi(-2y-1+M)t}{2N} = \sum_{x=0}^{M-1}\left(\frac{M-1}{2}-x\right)\cos\frac{\pi(-2x-1+M)t}{2N} \\ &= -\sum_{x=0}^{M-1}\left(x-\frac{M-1}{2}\right)\cos\frac{\pi(2x+1-M)t}{2N} = -B_c(t). \end{aligned}$$

Thus, it yields to $B_c(t) = 0$.

Appendix B. The proof of Lemma 2

From

$$A_s(t) = \sum_{x=0}^{M-1} \sin\frac{\pi(2x+1-M)t}{2N},$$

let $y = M - 1 - x$, then we have

$$\begin{aligned} A_s(t) &= \sum_{y=0}^{M-1} \sin \frac{\pi(2(M-1-y)+1-M)t}{2N} = \sum_{y=0}^{M-1} \sin \frac{\pi(-2y-1+M)t}{2N} \\ &= \sum_{x=0}^{M-1} \sin \frac{\pi(-2x-1+M)t}{2N} = -\sum_{x=0}^{M-1} \sin \frac{\pi(2x+1-M)t}{2N} = -A_s(t). \end{aligned}$$

It implies that $A_s(t) = 0$.

From

$$A'_c(t) = -\sum_{x=0}^{M-1} \frac{\pi(2x+1-M)}{2N} \sin \frac{\pi(2x+1-M)t}{2N} = -\frac{\pi}{N} B_s(t),$$

the closed form of $B_s(t)$ is given by

$$B_s(t) = -\frac{N}{\pi} A'_c(t) = -\frac{N}{\pi} \frac{\frac{\pi M}{2N} \sin \frac{\pi t}{2N} \cos \frac{\pi M t}{2N} - \frac{\pi}{2N} \cos \frac{\pi t}{2N} \sin \frac{\pi M t}{2N}}{\sin^2 \frac{\pi t}{2N}}.$$

References

[1] N. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, IEEE Trans. Comput. 23 (1974) 90–93.

[2] M. Anguh, Quadtree and symmetry in FFT computation of digital images, IEEE Trans. Signal Process. 45 (12) (1997) 2896–2899.

[3] K.L. Chung, J.G. Wu, Improved image compression using S-tree and shading approach, IEEE Trans. Commun. 48 (5) (2000) 748–751.

[4] E. Feig, S. Winograd, Fast algorithms for the discrete cosine transform, IEEE Trans. Signal Process. 40 (9) (1984) 2174–2193.

[5] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, Computer Graphics, Principle, and Practice, 2nd edn., Addison-Wesley, Reading, MA, 1990.

[6] H.S. Hou, A fast recursive algorithm for computing the discrete cosine transform, IEEE Trans. Acoust. Speech Signal Process. 35 (10) (1987) 1455–1461.

[7] B.G. Lee, A new algorithm to compute the discrete cosine transform, IEEE Trans. Acoust. Speech Signal Process. 32 (12) (1984) 1243–1245.

[8] P.Z. Lee, An efficient algorithm for the 2D discrete cosine transform, Signal Process. 55 (2) (1996) 221–239.

[9] W. Philips, On computing the FFT of digital images in quadtree format, IEEE Trans. Signal Process. 47 (7) (1999) 2059–2060.

[10] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, New York, 1990.

[11] H. Samet, Applications of Spatial Data Structures: Computer Graphics, Imaging Processing, and GIS, Addison-Wesley, New York, 1990.

[12] Z. Wang, Fast algorithms for the discrete W transform and for the discrete Fourier transform, IEEE Trans. Acoust. Speech Signal Process. 32 (4) (1984) 803–816.

[13] H.R. Wu, F.J. Paoloni, Comments on A 2D fast cosine transform algorithm based on Hou's approach, IEEE Trans. Signal Process. 39 (2) (1991) 544–546.

[14] H.R. Wu, Z. Man, Comments on Fast algorithms and implementation of 2D discrete cosine transform, IEEE Trans. Circuits Systems Video Technol. 8 (2) (1998) 128–129.