Digital Signal Processing 58 (2016) 20–25

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Two-dimensional discrete cosine transform on sliding windows

Chun-Su Park

Department of Digital Contents, Sejong University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Available online 21 July 2016

Keywords:
Discrete cosine transform
Sliding transform
Sliding window
Recursive algorithm
Two-dimensional algorithm

ABSTRACT

The discrete cosine transform (DCT) has been successfully used for a wide range of applications in digital signal processing. While there are efficient algorithms for implementing the DCT, its use becomes difficult in the sliding transform scenario where the transform window is shifted one sample at a time and the transform process is repeated. In this paper, a new two-dimensional sliding DCT (2-D SDCT) algorithm is proposed for fast implementation of the DCT on 2-D sliding windows. In the proposed algorithm, the DCT coefficients of the shifted window are computed by exploiting the recursive relationship between 2-D DCT outputs of three successive windows. The theoretical analysis shows that the computational requirement of the proposed 2-D SDCT algorithm is the lowest among existing 2-D DCT algorithms. Moreover, the proposed algorithm enables independent updating of each DCT coefficient.

1. Introduction

The discrete cosine transform (DCT) is an orthogonal transform that is very popular in modern digital signal processing. For highly correlated signals, the DCT provides similar performance to the Karhunen–Loeve transform (KLT) that is the optimal linear transform in terms of energy compaction. The DCT has a variety of applications in the fields of spectral analysis, speech and video processing, filter design, texture synthesis, etc. Due to its high complexity and time consuming requirements, the fast implementation of the DCT has been extensively investigated [1–5]. In particular, fast two-dimensional (2-D) DCT algorithms [6–13] were introduced for image and video applications such as tampered-image detection [14], image quality assessment [15,16], object detection [17], image retrieval [18], and watermarking [19].

In recent years, there has been a growing interest in sliding orthogonal transforms where the transform window is shifted one sample at a time and the transform process is repeated. While there are efficient algorithms for directly implementing orthogonal transforms, the computation of the sliding transform over a long sequence is very time consuming. Therefore, numerous algorithms have been developed for fast computation of sliding orthogonal transforms such as the discrete Fourier transform [20–22], the Walsh–Hadamard transform [23,24], and the DCT. Here, we consider the sliding transform algorithms designed for the DCT, which have been already applied to several practical applications including image denoising [25–28] and image restoration [29–31]. In recent studies [32,33], it was presented that the SDCT-based algorithms show relatively good performances for the applications.

SDCT algorithms can be categorized into two classes: first-order algorithms using two successive windows and second-order algorithms using three successive windows. In [34] and [35], the shift properties of the first-order were derived. The algorithm in [34] updates the DCT of the current window using both the DCT and the discrete sine transform (DST) of the previous window. Therefore, the DCT and the corresponding DST should be always updated even when only the DCT needs to be obtained. The complexity reduction of the algorithm is not satisfactory. In [35], the authors represent the DCT and the DST as the real part of complex functions. Then, the first-order recursive relationship between these complex functions was derived.

It was reported that further reduction in computational complexity can be achieved by exploiting the recursive relationship between three successive windows. The second-order shift properties of the DCT and the DST were derived in [36–39]. In [39], the second-order shift properties were well summarized and fast algorithms for performing sliding sinusoidal transforms were introduced. The second-order algorithms in [39] reduce the computational complexity by more than half as compared to the first-order algorithms in [34] and [35]. Moreover, the second-order algorithms enable independent updating of each DCT coefficient. This separable feature is highly beneficial for hardware implementation and multi-core processors. Recently, the generalize SDCT (gSDCT) algorithm [40] was introduced to handle the case where the window shifts multiple samples at a time. However, the computational requirement of the gSDCT is relatively high. It is worthwhile to note that all of these algorithms were basically designed to accelerate

E-mail address: cspark@sejong.ac.kr.

http://dx.doi.org/10.1016/j.dsp.2016.07.011
1051-2004/

the sliding transform process of one dimensional (1-D) input samples.

In this paper, we propose a 2-D SDCT algorithm for fast implementation of the DCT on 2-D sliding windows. This work is motivated by the previous work [39]. At first, we analyze the recursive relationship between 2-D DCT outputs of three successive windows. Based on the analysis, we propose a fast algorithm that obtains the 2-D DCT of the shifted window by performing 1-D DCT only once instead of 2-D DCT. The theoretical analysis shows that the computational requirement of the proposed 2-D SDCT algorithm is the lowest among existing 2-D DCT algorithms. Moreover, the output of the 2-D SDCT is mathematically equivalent to that of the traditional DCT at all sample positions.

The rest of this paper is organized as follows. In Section 2, the preliminary details of the sliding transform are given. In Section 3, we introduce the recursive relationship between 2-D DCT outputs of three successive windows. Section 4 presents a fast 1-D DCT algorithm that is required for implementing the proposed 2-D SDCT algorithm. Next, Section 5 illustrates the overall process and analyzes the computational requirement of the proposed algorithm. Finally, our conclusions are drawn in Section 6.

2. Preliminary

A sliding transform is based on the concept of short-time signal processing [41,42]. In the sliding transform, the transform is computed on a fixed-length window of the signal, which is continuously updated with new samples as the oldest ones are discarded. For simplicity of explanation, we consider the case where an $N \times N$ window is shifted in the horizontal direction. The other cases can be easily generalized.

Let us denote the (i, j)th sample of a 2-D input signal by $x(i, j)$. Further, let $X^{i,j}(u, v)$, $u, v = 0, 1, \ldots, N-1$, be the (u, v)th DCT coefficient at window position (i, j). There are four established types, DCT-I through DCT-IV, of the DCT which differ in the boundary conditions at the ends of the interval [43]. In this paper, we focus on the most commonly used form (DCT-II) and represent the sliding transform process at window position (i, j) as

$$X^{i,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(i+m, j+n) \cdot \cos\left(\frac{\pi(m+1/2)u}{N}\right)\cos\left(\frac{\pi(n+1/2)v}{N}\right) \tag{1}$$

where the normalization factor of the DCT is neglected for simplicity [39]. Let us define, respectively, symbols for representing the cosine and sine functions as follows

$$C_b^a = \cos\left(\frac{\pi(a+1/2)b}{N}\right) \tag{2}$$

and

$$S_b^a = \sin\left(\frac{\pi(a+1/2)b}{N}\right) \tag{3}$$

where $a, b = 0, 1, \ldots, N-1$. Then, the 2-D SDCT in (1) is rewritten as

$$X^{i,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(i+m, j+n) C_u^m C_v^n. \tag{4}$$

Using these notations, we will derive the recursive relationship between the DCT coefficients with the same spectral index of three successive windows and propose a fast 2-D SDCT algorithm in the next section.

3. Proposed sliding DCT algorithm for window size $N \times N$

According to (4), let us denote the (u, v)th DCT coefficient at window position $(i-1, j)$ by

$$X^{i-1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(i+m-1, j+n) C_u^m C_v^n. \tag{5}$$

The above equation can be divided into two parts as follows

$$X^{i-1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=1}^{N-1} x(i+m-1, j+n) C_u^m C_v^n + \sum_{n=0}^{N-1} x(i-1, j+n) C_u^0 C_v^n. \tag{6}$$

Then, using the property of the cosine function, we have

$$X^{i-1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=1}^{N-1} x(i+m-1, j+n) \times (C_u^{m-1}C_u^{1/2} - S_u^{m-1}S_u^{1/2}) C_v^n + \sum_{n=0}^{N-1} x(i-1, j+n) C_u^0 C_v^n \tag{7}$$

where

$$C_u^m = C_u^{m-1}C_u^{1/2} - S_u^{m-1}S_u^{1/2}. \tag{8}$$

Similarly, $X^{i+1,j}(u, v)$ can be expressed as

$$X^{i+1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-2} x(i+m+1, j+n) \times (C_u^{m+1}C_u^{1/2} + S_u^{m+1}S_u^{1/2}) C_v^n + \sum_{n=0}^{N-1} x(i+N, j+n) C_u^{N-1} C_v^n \tag{9}$$

where

$$C_u^m = C_u^{m+1}C_u^{1/2} + S_u^{m+1}S_u^{1/2}. \tag{10}$$

Next, we shift, respectively, the index m of summations in (7) and (9) by -1 and $+1$ and, after some manipulation, we have

$$X^{i-1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(i+m, j+n)(C_u^m C_u^{1/2} - S_u^m S_u^{1/2}) C_v^n - \sum_{n=0}^{N-1} x(i+N-1, j+n) C_u^N C_v^n + \sum_{n=0}^{N-1} x(i-1, j+n) C_u^0 C_v^n \tag{11}$$

and

$$X^{i+1,j}(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(i+m, j+n)(C_u^m C_u^{1/2} + S_u^m S_u^{1/2}) C_v^n - \sum_{n=0}^{N-1} x(i, j+n) C_u^{-1} C_v^n + \sum_{n=0}^{N-1} x(i+N, j+n) C_u^{N-1} C_v^n. \tag{12}$$