ELSEVIER

Signal Processing 68 (1998) 175–182

SIGNAL PROCESSING

On matrix factorizations for recursive pruned discrete cosine transforms

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

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

[b] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10764, Taiwan, ROC*

Received 27 August 1997; received in revised form 6 February 1998

Abstract

This paper presents two matrix factorizations for recursive pruned discrete cosine transforms. Both factorizations have lower computational complexity when compared to the method of El-Sharkawy and Eshmawy (1995). © 1998 Elsevier Science B.V. All rights reserved.

Zusammenfassung

Dieser Artikel behandelt zwei Methoden zur Matrizenfaktorisierung für rekursive reduzierte diskrete Cosinustransformationen. Beide Methoden haben einen geringeren Rechenaufwand als die Methode von El-Sharkawy und Eshmawy (1995). © 1998 Elsevier Science B.V. All rights reserved.

Résumé

Cet article présente deux méthodes de factorisation de matrices pour les transformations discrètes en cosinus élaguées récursives. Ces méthodes présentent toutes deux une complexité de calcul inférieure à celle de la méthode de El-Sharkawy et Eshmawy (1995). © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Computational complexity; Fast algorithm; Matrix factorization; Recursive pruned discrete cosine transform

1. Introduction

The recursive discrete cosine transforms (DCTs) [1,5] have many important applications in the area of image processing, communication, and digital signal processing. These applications include image

* Corresponding author. E-mail: klchung@cs.ntust.edu.tw. Prof. Chung was supported in part by the National Science Council of R.O.C. under contracts NSC87-2213-E011-001 and NSC87-2213-E011-003.

[1] Prof. Yan was supported in part by the National Science Council of R.O.C. under contract NSC87-2119-M002-006.

compression, filtering, feature extraction, teleconferencing, video phones, progressive image transmission, and so on. Due to the energy-packing capabilities, it approaches the statistically optimum Karhunen–Loeve transform for first-order Markov stationary random data [2]. In some applications, e.g. image compression, only the low-frequency DCT components should be kept, thus we only utilize a subset of input points or a subset of output points in order to decrease the computational complexity. This method is referred to as *pruning* [6,3].

Recently, El-Sharkawy and Eshmawy [3] presented an efficient matrix factorization for one-dimensional (1-D) recursive pruned DCT (RPDCT) which improves the result of Wang [6] significantly. Later, they [4] extended the proposed method from the 1-D domain to the 2-D domain successfully. Based on the result of [3], the motivation of this research is to present new matrix factorizations for RPDCT in order to gain some computational advantages.

Employing row operations and some addition rules for cosine functions, this paper presents two matrix factorizations for RPDCT. The proposed two factorizations are quite competitive with the result [3], but eliminate all the shift operations in one factorization and all but one in the other factorization.

The remainder of this paper is organized as follows. Section 2 introduces the work of El-Sharkawy and Eshmawy [3]. The proposed two matrix factorizations for RPDCT are presented in Section 3. Some concluding remarks are addressed in Section 4.

2. The work of El-Sharkawy and Eshmawy

For describing the proposed two matrix factorizations more clearly, in this section, the work of El-Sharkawy and Eshmawy [3] is reviewed in detail.

For $0 \leqslant i \leqslant n-1$, the Hadamard recurrence is given by $h_{2n}(2i) = h_n(i)$ and $h_{2n}(2i+1) = 2n-1-h_n(i)$ with the initial condition $h_1(0) = 0$, where the constraint $n = 2^k$ must be satisfied. From [6], the pruned DCT matrix is given by

$$C_n^{\mathrm{II}} = \begin{bmatrix} \sqrt{(1/n)}t_{0,0} & \sqrt{(1/n)}t_{0,1} & \cdots & \sqrt{(1/n)}t_{0,n-1} \\ \sqrt{(2/n)}t_{1,0} & \sqrt{(2/n)}t_{1,1} & \cdots & \sqrt{(2/n)}t_{1,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ \sqrt{(2/n)}t_{n-1,0} & \sqrt{(2/n)}t_{n-1,1} & \cdots & \sqrt{(2/n)}t_{n-1,n-1} \end{bmatrix},$$

where $t_{i,j} = \cos(2h_n(j)+1)i\pi/2n = \cos 2i\phi_j$ with $\phi_j = (2h_n(j)+1)\pi/4n$. For the pruned DCT, we want to compute $C_n^{\mathrm{II}}\boldsymbol{x}$, where $\boldsymbol{x}$ is the related input sequence with length n. Let

$$T(n) = \begin{bmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,n-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n-1,0} & t_{n-1,1} & \cdots & t_{n-1,n-1} \end{bmatrix},$$

then the following two steps can be used to compute $C_n^{\mathrm{II}}\boldsymbol{x}$:

Step 1. Compute $\boldsymbol{y} = T(n)\boldsymbol{x}$.

Step 2. Compute $\mathrm{diag}[\sqrt{1/n}, \sqrt{2/n}, \sqrt{2/n}, \ldots, \mu\sqrt{2/n}]\boldsymbol{y}$.

The above two steps need $\mathrm{O}(n^2)$ multiplications and additions since they mainly concern a matrix–vector multiplication.

In [3], El-Sharkawy and Eshmawy presented a faster method to reduce the above time bound to $\mathrm{O}(n \log n)$ multiplications and additions.

Let $P(n)$ be an $n\times n$ permutation matrix formed by reordering the rows of the identity matrix so that the following equality holds: $P(n)(0,1,2,\dots,n-2,n-1)^t=(0,2,\dots,n-2,1,3,\dots,n-1)^t$. The previous matrix $T(n)$ is reordered by exchanging rows and columns of the matrix, then the resultant matrix is decomposed into four quadrants [3] as shown below:

$$P(n)T(n)P(n)^{\mathrm{T}}=\begin{bmatrix} T_{ee}(\frac{n}{2}) & T_{eo}(\frac{n}{2}) \\ T_{oe}(\frac{n}{2}) & T_{oo}(\frac{n}{2}) \end{bmatrix}, \tag{1}$$

where

$$T_{ee}\left(\frac{n}{2}\right)=\begin{bmatrix} t_{0,0} & t_{0,2} & \cdots & t_{0,n-2} \\ t_{2,0} & t_{2,2} & \cdots & t_{2,n-2} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n-2,0} & t_{n-2,2} & \cdots & t_{n-2,n-2} \end{bmatrix}, \quad T_{eo}\left(\frac{n}{2}\right)=\begin{bmatrix} t_{0,1} & t_{0,3} & \cdots & t_{0,n-1} \\ t_{2,1} & t_{2,3} & \cdots & t_{2,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n-2,1} & t_{n-2,3} & \cdots & t_{n-2,n-1} \end{bmatrix},$$

$$T_{oe}\left(\frac{n}{2}\right)=\begin{bmatrix} t_{1,0} & t_{1,2} & \cdots & t_{1,n-2} \\ t_{3,0} & t_{3,2} & \cdots & t_{3,n-2} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n-1,0} & t_{n-1,2} & \cdots & t_{n-1,n-2} \end{bmatrix} \quad \text{and} \quad T_{oo}\left(\frac{n}{2}\right)=\begin{bmatrix} t_{1,1} & t_{1,3} & \cdots & t_{1,n-1} \\ t_{3,1} & t_{3,3} & \cdots & t_{3,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n-1,1} & t_{n-1,3} & \cdots & t_{n-1,n-1} \end{bmatrix}.$$

From the definitions of $t_{i,j}$ and the Hadamard recurrence, we have

$$t_{i,2j}=\cos(2h_n(2j)+1)\mathrm{i}\frac{\pi}{2n}=\cos(2h_{n/2}(j)+1)\mathrm{i}\frac{\pi}{2n}=\cos\mathrm{i}\phi'_j,$$

where $\phi'_j=(2h_{n/2}(j)+1)\pi/2n$. Similarly, we have

$$t_{i,2j+1}=\cos(2h_n(2j+1)+1)\mathrm{i}\frac{\pi}{2n}=\cos(2(n-1-h_{n/2}(j))+1)\mathrm{i}\frac{\pi}{2n}$$

$$=\cos\mathrm{i}\pi-(2h_{n/2}(j))+1)\mathrm{i}\frac{\pi}{2n}=(-1)^{\mathrm{i}}\cos\mathrm{i}\phi'_j.$$

Then it follows that

$$t_{2i,2j}=t_{2i,2j+1}=\cos 2\mathrm{i}\phi'_j$$

and

$$t_{2i+1,2j}=-t_{2i+1,2j+1}=\cos(2i+1)\phi'_j.$$

It is not hard to verify that $T_{ee}(n/2)=T_{eo}(n/2)=T(n/2)$ and $T_{oo}(n/2)=-T_{oe}(n/2)$.

For analyzing the computational complexity, let $m=n/2$. We then have

$$T_{ee}(m)=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos 2\phi'_0 & \cos 2\phi'_1 & \cdots & \cos 2\phi'_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \cos(2m-2)\phi'_0 & \cos(2m-2)\phi'_1 & \cdots & \cos(2m-2)\phi'_{m-1} \end{bmatrix}$$

and

$$T_{oe}(m)=\begin{bmatrix}\cos\phi'_0 & \cos\phi'_1 & \cdots & \cos\phi'_{m-1}\\ \cos 3\phi'_0 & \cos 3\phi'_1 & \cdots & \cos 3\phi'_{m-1}\\ \vdots & \vdots & \cdots & \vdots\\ \cos(2m-1)\phi'_0 & \cos(2m-1)\phi'_1 & \cdots & \cos(2m-1)\phi'_{m-1}\end{bmatrix}.$$

The matrix $T_{oe}(m)$ has the following matrix factorization:

$$T_{oe}(m)=A(m)T_{ee}(m)B(m),$$

where

$$A(m)=\begin{bmatrix}1 & 0 & 0 & 0 & \cdots & 0\\ -1 & 2 & 0 & 0 & \cdots & 0\\ 1 & -2 & 2 & 0 & \cdots & 0\\ -1 & 2 & -2 & 2 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots\\ -1 & 2 & -2 & 2 & \cdots & 2\end{bmatrix}_{m\times m}$$

and

$$B(m)=\mathrm{diag}[\cos\phi'_0,\cos\phi'_1,\cos\phi'_2,\ldots,\cos\phi'_{m-1}].$$

From Eq. (1) and $T_{oe}(m)=A(m)T_{ee}(m)B(m)$, it is rather straightforward that

$$T(2m)=P(2m)^{\mathrm{T}}\begin{bmatrix}T_{ee}(m) & T_{eo}(m)\\ T_{oe}(m) & T_{oo}(m)\end{bmatrix}P(2m)=P(2m)^{\mathrm{T}}\begin{bmatrix}T(m) & T(m)\\ A(m)T(m)B(m) & -A(m)T(m)B(m)\end{bmatrix}P(2m).$$

We now want to compute $\boldsymbol{y}=T(2m)\boldsymbol{x}$, where $\boldsymbol{x}$ is the input sequence with length $2m$. First, we compute

$$\boldsymbol{v}=\begin{bmatrix}\boldsymbol{v}_1\\ \boldsymbol{v}_2\end{bmatrix}=P(2m)\boldsymbol{x}.$$

Then, we compute

$$\begin{bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{bmatrix}=\begin{bmatrix}T(m) & T(m)\\ A(m)T(m)B(m) & -A(m)T(m)B(m)\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1\\ \boldsymbol{v}_2\end{bmatrix}=\begin{bmatrix}T(m)(\boldsymbol{v}_1+\boldsymbol{v}_2)\\ A(m)T(m)B(m)(\boldsymbol{v}_1-\boldsymbol{v}_2)\end{bmatrix}.$$

Finally, we compute

$$\boldsymbol{y}=P(2m)^{\mathrm{T}}\begin{bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{bmatrix}.$$

Equivalently, the procedure [3] consisting of the following six steps for computing $\boldsymbol{y}=T(2m)\boldsymbol{x}$ is shown below:

Step 1: Compute $\boldsymbol{v}=P(2m)\boldsymbol{x}$.
Step 2: Compute $\boldsymbol{w}_1=\boldsymbol{v}_1+\boldsymbol{v}_2$ and $\boldsymbol{w}_2=\boldsymbol{v}_1-\boldsymbol{v}_2$.
Step 3: Compute $\boldsymbol{w}'_2=B(m)\boldsymbol{w}_2$.
Step 4: Compute $\boldsymbol{z}_1=T(m)\boldsymbol{w}_1$ and $\boldsymbol{z}'_2=T(m)\boldsymbol{w}'_2$.
Step 5: Compute $\boldsymbol{z}_2=A(m)\boldsymbol{z}'_2$.
Step 6: Compute $\boldsymbol{y}=P(2m)^{\mathrm{T}}\boldsymbol{z}'$, where $\boldsymbol{z}'=\begin{bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{bmatrix}$.

In the next section, two new matrix factorizations for $T_{oe}(m) = A(m)T_{ee}(m)B(m)$ are presented. For the first proposed factorization, the computational complexity required in Step 5 of the above procedure can be reduced from (($m-1$) shift operations and ($m-1$) subtractions) to (one shift and ($m-1$) subtractions), while preserving the same bound in the remaining steps. For the second proposed factorization, the computational complexity required in Step 5 in the above procedure can be reduced to ($m-1$) additions, while still preserving the same bound in the remaining steps.

3. Two new matrix factorizations

Let

$$T_{oe}(m) = \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{m-1} \end{bmatrix} \quad \text{and} \quad T_{ee}(m) = \begin{bmatrix} \boldsymbol{r}'_0 \\ \boldsymbol{r}'_1 \\ \vdots \\ \boldsymbol{r}'_{m-1} \end{bmatrix},$$

where

$$\boldsymbol{r}_k = (\cos(2k+1)\phi'_0, \cos(2k+1)\phi'_1, \ldots, \cos(2k+1)\phi'_{m-1}),$$

$$\boldsymbol{r}'_k = (\cos 2k\phi'_0, \cos 2k\phi'_1, \ldots, \cos 2k\phi'_{m-1}),$$

for $k = 0,1,\ldots,m-1$. From $\phi'_j = (2h_m(j)+1)\pi/4m$, it follows that $\boldsymbol{r}'_m = (\cos 2m\phi'_0, \cos 2m\phi'_1, \ldots, \cos 2m\phi'_{m-1}) = (0,0,\ldots,0,0)$.

We add the ($k-1$)th row vector in $T_{oe}(m)$ to the kth row vector for $1 \leqslant k \leqslant m-1$, then we have

$$\boldsymbol{r}_{k-1} + \boldsymbol{r}_k = (\cos(2k-1)\phi'_0 + \cos(2k+1)\phi'_0, \cos(2k-1)\phi'_1 + \cos(2k+1)\phi'_1, \ldots,$$
$$\cos(2k-1)\phi'_{m-1} + \cos(2k+1)\phi'_{m-1}).$$

From the trigonometric addition identity:

$$\cos(2k-1)\phi'_j + \cos(2k+1)\phi'_j = 2\cos\phi'_j \cos 2k\phi'_j = \alpha_j \cos 2k\phi'_j,$$

it follows that

$$\boldsymbol{r}_{k-1} + \boldsymbol{r}_k = (\alpha_0 \cos 2k\phi'_0, \alpha_1 \cos 2k\phi'_1, \ldots, \alpha_{m-1}\cos 2k\phi'_{m-1}),$$

where $\alpha_j = 2\cos\phi'_j$.

By the same argument, we add the kth row vector in $T_{ee}(m)$ to the ($k+1$)th row vector for $0 \leqslant k \leqslant m-2$. From the trigonometric addition identity:

$$\cos 2k\phi'_j + \cos(2k+2)\phi'_j = 2\cos\phi'_j \cos(2k+1)\phi'_j = \alpha_j \cos(2k+1)\phi'_j,$$

we have

$$\boldsymbol{r}'_k + \boldsymbol{r}'_{k+1} = (\alpha_0 \cos(2k+1)\phi'_0, \alpha_1 \cos(2k+1)\phi'_1, \ldots, \alpha_{m-1}\cos(2k+1)\phi'_{m-1}).$$

We then have

$$\begin{bmatrix} 2\boldsymbol{r}_0 \\ \boldsymbol{r}_0 + \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{m-2} + \boldsymbol{r}_{m-1} \end{bmatrix} = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{m-1} \\ \alpha_0 \cos 2\phi'_0 & \alpha_1 \cos 2\phi'_1 & \cdots & \alpha_{m-1}\cos 2\phi'_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_0 \cos(2m-2)\phi'_0 & \alpha_1 \cos(2m-2)\phi'_1 & \cdots & \alpha_{m-1}\cos(2m-2)\phi'_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos 2\phi'_0 & \cos 2\phi'_1 & \cdots & \cos 2\phi'_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \cos(2m-2)\phi'_0 & \cos(2m-2)\phi'_1 & \cdots & \cos(2m-2)\phi'_{m-1} \end{bmatrix} \begin{bmatrix} \alpha_0 & & & \\ & \alpha_1 & & \\ & & \cdots & \\ & & & \alpha_{m-1} \end{bmatrix}$$

$$= T_{ee}(m)D(m)$$

and from $\boldsymbol{r}'_m = (0,0,\dots,0,0)$, it yields

$$\begin{bmatrix} \boldsymbol{r}'_0 + \boldsymbol{r}'_1 \\ \vdots \\ \boldsymbol{r}'_{m-2} + \boldsymbol{r}'_{m-1} \\ \boldsymbol{r}'_{m-1} + \boldsymbol{r}'_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}'_0 + \boldsymbol{r}'_1 \\ \vdots \\ \boldsymbol{r}'_{m-2} + \boldsymbol{r}'_{m-1} \\ \boldsymbol{r}'_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_0 \cos \phi'_0 & \alpha_1 \cos \phi'_1 & \cdots & \alpha_{m-1} \cos \phi'_{m-1} \\ \alpha_0 \cos 3\phi'_0 & \alpha_1 \cos 3\phi'_1 & \cdots & \alpha_{m-1} \cos 3\phi'_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_0 \cos(2m-1)\phi'_0 & \alpha_1 \cos(2m-1)\phi'_1 & \cdots & \alpha_{m-1} \cos(2m-1)\phi'_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \cos \phi'_0 & \cos \phi'_1 & \cdots & \cos \phi'_{m-1} \\ \cos 3\phi'_0 & \cos 3\phi'_1 & \cdots & \cos 3\phi'_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \cos(2m-1)\phi'_0 & \cos(2m-1)\phi'_1 & \cdots & \cos(2m-1)\phi'_{m-1} \end{bmatrix} \begin{bmatrix} \alpha_0 & & & \\ & \alpha_1 & & \\ & & \cdots & \\ & & & \alpha_{m-1} \end{bmatrix}$$

$$= T_{oe}(m)D(m),$$

where

$$D(m) = \mathrm{diag}[\alpha_0,\alpha_1,\dots,\alpha_{m-1}] = \mathrm{diag}[2\cos\phi'_0, 2\cos\phi'_1, \dots, 2\cos\phi'_{m-1}].$$

In summary, we have the following two important equations:

$$\begin{bmatrix} 2\boldsymbol{r}_0 \\ \boldsymbol{r}_0 + \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{m-2} + \boldsymbol{r}_{m-1} \end{bmatrix} = T_{ee}(m)D(m) \tag{2}$$

and

$$\begin{bmatrix} \boldsymbol{r}'_0 + \boldsymbol{r}'_1 \\ \vdots \\ \boldsymbol{r}'_{m-2} + \boldsymbol{r}'_{m-1} \\ \boldsymbol{r}'_{m-1} \end{bmatrix} = T_{oe}(m)D(m). \tag{3}$$

Let

$$L(m)=\begin{bmatrix}2&0&0&0&\cdots&0\\1&1&0&0&\cdots&0\\0&1&1&0&\cdots&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\0&\cdots&0&1&1&0\\0&\cdots&0&0&1&1\end{bmatrix}_{m\times m}\quad\text{and}\quad U(m)=\begin{bmatrix}1&1&0&0&\cdots&0\\0&1&1&0&\cdots&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\0&\cdots&0&1&1&0\\0&\cdots&0&0&1&1\\0&\cdots&0&0&0&1\end{bmatrix}_{m\times m}.$$

Eqs. (2) and (3) can be rewritten as

$$L(m)T_{oe}(m)=L(m)\begin{bmatrix}\boldsymbol{r}_0\\\boldsymbol{r}_1\\\vdots\\\boldsymbol{r}_{m-1}\end{bmatrix}=\begin{bmatrix}2\boldsymbol{r}_0\\\boldsymbol{r}_0+\boldsymbol{r}_1\\\vdots\\\boldsymbol{r}_{m-2}+\boldsymbol{r}_{m-1}\end{bmatrix}=T_{ee}(m)D(m)$$

and

$$U(m)T_{ee}(m)=U(m)\begin{bmatrix}\boldsymbol{r}'_0\\\vdots\\\boldsymbol{r}'_{m-2}\\\boldsymbol{r}'_{m-1}\end{bmatrix}=\begin{bmatrix}\boldsymbol{r}'_0+\boldsymbol{r}'_1\\\vdots\\\boldsymbol{r}'_{m-2}+\boldsymbol{r}'_{m-1}\\\boldsymbol{r}'_{m-1}\end{bmatrix}=T_{oe}(m)D(m).$$

As a result, two new factorizations for $T_{oe}(m)$ are derived as shown below:

$$T_{oe}(m)=L^{-1}(m)T_{ee}(m)D(m), \tag{4}$$

$$T_{oe}(m)=U(m)T_{ee}(m)D^{-1}(m). \tag{5}$$

From Eq. (4), computing Step 5 is equal to solving the bidiagonal linear system $L(m)\boldsymbol{z}_2=\boldsymbol{z}'_2$, and it takes one shift operation and $m-1$ subtractions. From Eq. (5), it takes $m-1$ additions to compute Step 5.

Recall that the computational complexity required in Step 5 of the above procedure mentioned in Section 2 is $(m-1)$ shift operations and $(m-1)$ subtractions. The computational complexity required in Step 5 of any one of the proposed two methods is lower than the that of in the above procedure, while preserving the same bound in remaining steps. Table 1 illustrates the computational complexity required in Step 5 among the three methods.

In Table 1, if we assume that the computational load of one subtraction is equal to that of one addition, the second method is the fastest among the three methods.

Table 1
Computational complexity comparison for Step 5

	Shift operations	Subtractions	Additions
[3]	$m-1$	$m-1$	0
First method	1	$m-1$	0
Second method	0	0	$m-1$

4. Concluding remarks

We have presented two matrix factorizations to improve some steps in the procedure of El-Sharkawy and Eshmawy [3] for recursive pruned discrete cosine transforms. How to design efficient matrix factorizations to handle the case $n \neq 2^k$ is our future research topic. Instead of using trigonometric identities to derive the proposed matrix factorizations, how to just change some multiplicators to derive the same results is another interesting research issue.

Acknowledgements

The authors are indebted to the two reviewers and Prof. Murat Kunt for making some valuable suggestions and corrections that lead to the improved version of the paper.

References

[1] N. Ahmed, K.R. Rao, Orthogonal Transforms in Digital Signal Processing, Springer, New York, 1975.
[2] R.J. Clack, Relation between the Karhunen–Loeve and cosine transform, Proc. IEE. Part F 128 (November 1981) 359–360.
[3] M. El-Sharkawy, W. Eshmawy, A fast recursive pruned discrete cosine transform, Digital Signal Process. 5 (1995) 140–148.
[4] M. El-Sharkawy, W. Eshmawy, A fast 8×8 pruned DCT algorithm, Digital Signal Process. 6 (1996) 145–154.
[5] K.R. Rao, P. Yip, Discrete Cosine Transform: Algorithms, Advantages, and Applications, Academic Press, New York, 1990.
[6] Z. Wang, Pruning the fast discrete cosine transform, IEEE Trans. Commun. 39 (1991) 640–643.