**Prime Factor Algorithm of Discrete Cosine Transform**

Guoan Bi and Yonghong Zeng,
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore 639798

e-mail: egbi, eyhzeng@ntu.edu.sg

**ABSTRACT**
Prime factor fast algorithms are computationally efficient for various discrete transforms. However, they generally need an index mapping process to convert one-dimensional input sequence into a two-dimensional array, which results in a substantially computational overhead and an irregular computational structure. This letter proposes a simple mapping method for the PFA of the discrete cosine transform.

1. INTRODUCTION
The discrete cosine transform (DCT) is an important tool in digital signal processing. Many fast algorithms were reported to reduce the computational and structural complexity ([1-8], for example). The prime factor algorithm (PFA) [3-8] was considered to be computationally efficient because twiddle factors are not needed. The basic principle of the PFA is to convert the one-dimensional input sequence into a two-dimensional one. Therefore, a mapping process is needed to convert the original one-dimensional input sequence into a two-dimensional array. The mapping process is generally undesirable because it complicates the computational structure and requires modulo and other arithmetic operations, which imposes a substantially computational overhead. By using table or matrix manipulation, several methods were proposed to deal with the mapping procedure [4, 6-8]. These methods are not straightforward and difficult to use particularly for large values of prime factors. This letter attempts to minimize the computation overhead by a simple and general mapping procedure.

2. ALGORITHM
The type-II DCT of sequence \( x(n) \), \( n=0,1,...,N-1 \), is defined by
\[
X(k) = \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)k}{2N} \quad 0 \leq k \leq N-1
\] (1)

If \( N = p \times q \) and \( p \) and \( q \) are mutually prime, we have
\[
X(\mu k + q m) = \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)(\mu k + q m)}{2N} = A(k, m) - B(\mu k, m)
\] (2)

where
\[
A(k, m) = \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)m}{2p} \cos \frac{(2n+1)k}{2q}
\] (3)

and
\[
B(\mu k, m) = \mu \sum_{n=0}^{N-1} x(n) \sin \frac{(2n+1)m}{2p} \sin \frac{(2n+1)k}{2q}
\] (4)

where for (2) to (4), \( k=0\ldots q-1, m=0\ldots p-1 \), and \( \mu \) is defined by
\[
\mu = \begin{cases} 1 \text{ if } pk +mq < N \\ -1 \text{ if } pk +mq > N 
\end{cases}
\] (5)

Because \( \mu k + q m \) in (2) can be negative, the absolute value \( |\mu k + q m| \) is used as the output indices. Furthermore, (3) and (4) are related by
\[
B(\mu k, m) = \mu A(q - k, p - m) \quad B(k, 0) = 0
\] (6)

which indicates that (4) can be obtained from (3). Therefore, the length-\( N \) DCT can be obtained if \( A(k, m) \) is available. Let us express \( A(k, m) \) to be
\[
A(k, m) = \sum_{n=0}^{N-1} \left[ \begin{array}{l}
\sum_{i=-\lfloor p/2 \rfloor}^{\lfloor p/2 \rfloor} y(i, n) \cos \pi (4i q + 2n + 1) m \\
\sum_{i=-\lfloor p/2 \rfloor}^{\lfloor p/2 \rfloor} y(i, n) \cos \pi (4i q - 2n - 1) m
\end{array} \right] \cos \frac{(2n+1)k}{2q}
\] (7)

The summation inside the braces can be converted into a length-\( p \) DCT as follows. By using the property \( \cos(a) = \cos(-a) \) in (7), the two terms in the braces can be rewritten into a close form
\[
\sum_{i=-\lfloor p/2 \rfloor}^{\lfloor p/2 \rfloor} y(i, n) \cos \pi (4i q + 2n + 1) m = \frac{2}{p} \sum_{i=0}^{i \geq 0} \cos \frac{\pi (4i q - 4n - 1) m}{2p} \cos \frac{(2n+1)k}{2q}
\] (8)

where
\[
y(i, n) = \begin{cases} x(2i q + n) \quad i \geq 0 \\
x(-2i q + n - 1) \quad i < 0
\end{cases}
\] (9)

To convert (8) into a length-\( p \) DCT, it is necessary to find \( p \) values of \( l \) for each \( n \)
\[
\cos \frac{\pi (2l + 1) m}{2p} = \cos \frac{\pi (4i q + 2n + 1) m}{2p}
\]

Without presenting detailed proof, this relation can be equivalently expressed by
\[
2l + 1 = \begin{cases} \| 4i q + 2n + 1 \| - 4p \quad \text{if } |4i q + 2n + 1| > 2p \\
|4i q + 2n + 1| \quad \text{if } |4i q + 2n + 1| < 2p
\end{cases}
\]

\[
n = 0,1...,q-1; \quad i = -(p-1)/2,0,...,(p-1)/2, \quad l = 0,1,...,p-1
\] (10)

Therefore, (8) can be converted to
\[ \sum_{l=0}^{p-1} y(l, n) \cos \frac{\pi(2l+1)m}{2p} \] (11)

where \( l \) is derived from (10), and \( y(l, n) \) is a two-dimensional array converted from the original input sequence \( x(n) \) according to (9) and (10). Therefore, the length-\( N \) DCT is decomposed into \( p \) length-\( q \) and \( q \)-length-\( p \) DCTs. Figure 1 shows an example for a length-15 DCT. The decomposition does not require twiddle factors, and subtractions are needed in (2) to form the final outputs. Thus, the number of multiplications needed by the proposed algorithm is

\[ M(N) = p M(q) + q M(p) \] (12)

and the number of additions is

\[ A(N) = p A(q) + q A(p) + (p-1)(q-1) \] (13)

Compared to the algorithms in [6, 7], the proposed one requires the same number of arithmetic operations (addition plus multiplication) and a similar input/output index mapping. However, the proposed algorithm reduces about \( (p+1)(q-1) \) arithmetic operations compared to that in [4].

An important issue of the prime-factor algorithm is the realization of the mapping process so that its overhead does not offset the savings achieved by eliminating twiddle factors. A Ruritanian mapping was used in the form of tables [6], which was tedious and difficult to use for large prime factors. Because of the difficulties in realising the method of tabulation by a computer subroutine, a large number of modulo and other arithmetic operations is necessary for software implementation according to the procedures listed in [7]. For the proposed decomposition, a very simple and straightforward method can be utilised, as shown in Figure 2 for an example of a length-15 DCT (\( p=5 \) and \( q=3 \)). In Figure 2, \( 2l+1 \) and \( n \) are the indexing variables used in (11), the numbers inside the array are the original input indices, and the arrow indicates that the next input index (pointed by the arrow) is achieved by increment by one operation from the index from which the arrow leaves.

Table 1 shows a different presentation of the mapping process. Let \( i, 0 \leq i < p \), and \( j, 0 \leq j < q \), be the two-dimensional array indices, and \( n, 0 \leq n < N \), be the index of the original input sequence. In general, the mapping process can be performed by the following step. For each \( n, 0 \leq n < N \),

(i). starting from 0, the index \( i \) of the two-dimensional array is incremented by one until \( p-1 \), then from \( p-1 \), index \( i \) is decremented by one until to 0. This process is repeated;

(ii). index \( j \) is generated in the same way as \( i \) except the range is from 0 to \( q-1 \) for increment and from \( q-1 \) to 0 for decrement;

Therefore, each column in Table 1 specifies the relation between \((i, j)\) and \( n \). Appendix A illustrates the subroutine of the mapping process which needs only addition/subtraction and comparison.

In-place computation can also be realised because the inputs of each computational block (the length-5 DCT in Figure 1, for example) are no longer needed for further computation and the associated memory space can be used for the outputs of the computational block. The transform outputs are also to be arranged into a natural order. The expression \([yk+k+m] \) is used to relate each computed output to indices \( k, k=0,...,q-1 \), and \( m, m=0,..., p-1 \). This process can be easily implemented with operations of comparison, addition and subtraction.

3. Conclusion

An efficient index mapping for converting a one-dimensional input sequence into a two-dimensional array is presented for a prime factor algorithm of the discrete cosine transform. The mapping method is straightforward and general for any factors that are mutually prime. It reduces the computational overhead and avoids irregularity of the computational structure. Thus, the overall computational and implementation costs of the proposed prime factor algorithm can be minimized.

Appendix Subroutine of input index mapping

Input_mapping (p, q, index)
\[
\begin{align*}
\text{i} &= \text{i}+\text{k1}; \ j=\text{j}+\text{s1}; \\
\text{if} (\text{i}=\text{k2}) &\quad \text{/* i and j are the new array indices */} \\
\quad &\quad \text{if} (\text{k2}==\text{eq}) \quad \{ \text{i=}\text{i-1}; \text{k2=}\text{-1}; \text{k1=}\text{-1}; \} \\
\quad &\quad \text{else if} (\text{k2}==\text{-1}) \quad \{ \text{i=}\text{i+1}; \text{k2=}\text{q}; \text{k1=}\text{1}; \} \\
\quad &\quad \text{if} (\text{j}=\text{s2}) \\
\quad &\quad \quad \{ \text{j=}\text{j-1}; \text{s2=}\text{-1}; \text{s1=}\text{-1}; \} \\
\quad &\quad \text{else if} (\text{s2}==\text{-1}) \quad \{ \text{j=}\text{j+1}; \text{s2=}\text{p}; \text{s1=}\text{1}; \} \\
\quad &\quad \text{index}[\text{i}][\text{j}]=\text{n}; \\
\quad &\quad \text{/*index contains relation among i, j and n*/} \\
\quad &\quad \text{return(0);} 
\end{align*}
\]

References


[5]. D. C. Kar and V. V. B. Tao, “On the prime factor decomposition algorithm for the discrete sine

![Signal flow graph of DCT computation (p x q=15)](image)

*Figure 1* Signal flow graph of DCT computation (p x q=15)

![Input mapping graph for p=5 and q=3](image)

*Figure 2* Input mapping graph for p=5 and q=3

| Table I Mapping the 1 D input sequence into a 2 D array |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i=0 | 1 | 2 | 2 | 1 | 0 | 0 | 1 | 2 | 2 | 1 | 0 | 0 |
| j=0 | 1 | 2 | 3 | 4 | 4 | 3 | 2 | 1 | 0 | 0 | 1 | 2 |
| n=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |