FAST MULTIPLIERLESS APPROXIMATION OF THE DCT

Truc D. Tran

ECE Department, The Johns Hopkins University, Baltimore, MD 21218
ttran@ece.jhu.edu

ABSTRACT

In this paper, we present a fast biorthogonal block transform called binDCT that can be implemented using only shift and add operations. The transform is based on a VLSI-friendly lattice structure which robustly enforces both linear phase and perfect reconstruction properties. The lattice coefficients are parameterized as a series of dyadic shift and add operations. The transform is based on a discrete-time convolution called binDCT that can be implemented using only shift-and-add operations. The new transform coefficients as well as the ability to map integers to integers. The new transform in both lossy and lossless image coding yields very competitive results comparing to the performance of the original floating-point DCT.

1. INTRODUCTION

Block transforms have long found application in image and video coding. The current image compression standard JPEG [1] as well as many high-performance video coding standards such as MPEG and H.263 all employ the 8 × 8 discrete cosine transform (DCT) at its transformation stage. From a statistical signal processing standpoint, the DCT is a robust approximation to the optimal discrete-time Karhunen-Loève transform (KLT) of a first-order Gauss-Markov process with a positive correlation coefficient ρ when ρ → 1 [2].

The KLT is optimal in the energy compaction sense, i.e., among unitary transforms, the KLT packs signal energy into the fewest number of coefficients. However, the KLT is signal-dependent, therefore, computationally complex and expensive. The DCT has proven to be a much better alternative in practice: it is signal independent, it has linear phase, real coefficients, and fast algorithms.

Exploiting the symmetry of the basis functions, the DCT transform matrix Π can be factored into a series of ±1 butterflies and rotation angles as illustrated in Figure 1. This factorization results in one of the fastest DCT implementations known up to date [2]. Eight DCT coefficients X[i] can be computed using 13 multiplications and 29 additions. Note that all resulting coefficients X[i] have been scaled up by a factor of 2.

Despite all of the nice attributes mentioned above, there is still room for improvement. The DCT is a floating-point transform. It cannot map integers to integers losslessly. More importantly, floating-point implementations in hardware are slow, require too much space, and consume too much power. Most practical implementations of the DCT are based on integer arithmetic by scaling up the floating-point multipliers by very large factors. Certainly, this ad-hoc approximation method still has problems: it does not reduce the complexity much and it has truncation errors.

In this paper, we present a novel block transform that not only possesses the integer mapping capability but also has dyadic-rational coefficients that lead to an elegant implementation utilizing only shift-and-add operations.

2. GENERAL SOLUTION

2.1. General Factorization

From a filter bank standpoint, the M × M DCT is the most basic M-channel linear phase paraunitary filter bank (LPPUFB): all M filters having the same length M. Its polyphase matrix has order 0 (independent of z) and can be written in the following form:

\[ E_0 = \begin{bmatrix} U & UJ \\ VJ & -V \end{bmatrix} = \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} I & J \\ J & -I \end{bmatrix}. \]  

(1)

It is clear that \( E_0 \) is orthogonal if and only if \( U \) and \( V \) are orthogonal. For \( E_0 \) to represent the DCT, we need two special orthogonal matrices. However, any choice of orthogonal \( U \) and \( V \) does result in an M-channel M-tap LPPUFB. In the more general biorthogonal case, \( E_0 \) must...
be invertible. From the factorization above, it is clear that $E_0$ is invertible if and only if $U$ and $V$ are invertible. In other words, the factorization in Eq. (1) covers the complete class of all $M$-channel $M$-tap biorthogonal LPFBs as long as the free-parameter matrices $U$ and $V$ are invertible. The general structure is depicted in Figure 2.

2.2. Matrix Parameterization via Lifting Steps

The challenge is how to characterize these invertible $U$ and $V$ matrices using the fewest number of independent parameters. It is well-known that every $N \times N$ orthogonal matrix can be factored into $\frac{N(N-1)}{2}$ rotations. We shall now establish a similar result for invertible matrices.

**Theorem:** Every $N \times N$ invertible matrix can be completely characterized by $N(N - 1)$ shears, $N$ diagonal scaling factors, and possibly a permutation matrix.

The detailed proof is presented in the journal version of the paper [3]. It is not too difficult to see how one can systematically factor any invertible matrix using permutation, diagonal scaling, and shearing. Firstly, the invertible matrix of interest is permuted so that we have all non-zero elements on the diagonal. Next, all of the diagonal elements can then be scaled to $1$'s. Finally, the Gauss-Jordan elimination process can be applied to each non-diagonal element, turning it to zero. This is a shear operation, also called a lifting step in the context of this paper.

![Figure 3. Parameterization of an invertible matrix via the lifting steps and scaling factors.](image)

The parameterization of an arbitrary invertible matrix is illustrated in Figure 3 (drawn for $N = 4$). Back to our general $M$-channel biorthogonal block transform with linear phase basis functions, the transform can be proven to consist of $\left(\frac{N^2}{2} - M\right)$ lifting steps $l_i$ and $M$ diagonal scaling factors $a_i$. If we restrict the determinants of the matrices to $\pm 1$, the $M$ scaling factors $a_i$ can be converted into lifting steps as well. Typically, these scaling factors can be folded into the quantization stepsizes of the encoder.

2.3. Lifting Steps

The lifting step and its versatility in constructing fast transforms that can map integers to integers are demonstrated in Figure 4. First of all, inverting a lifting step is simple: just inverting its polarity when the diagonal elements of the matrix are unity with the same polarity (both +1 or -1). In the case that the diagonal elements have opposite polarity, the inverse of the lifting step is simply itself.

Secondly, if a floor (or round, or ceiling) operator is placed in each lifting step, our transform can now map integers to integers with perfect reconstruction. Thirdly, if the lifting step is chosen to be dyadic (i.e., a rational that can be written in the form of $\frac{k}{m}$, $k, m \in \mathbb{Z}$), the nonlinear operation can be incorporated into the division using binary bit shift. Division by $\frac{1}{m}$ followed by a truncation is equivalent to a right binary shift by $m$ places. The numerator $k$ can be easily implemented using bit shift and add operations as well. Hence, every multiplierless transform can be constructed using this method.

![Figure 4. Example of a lifting step and its implementations. (a) Original lifting step. (b) An approximation that can map integers to integers. (c) A shift-and-add approximation.](image)

3. THE MULTIPLIERLESS BINDCT

This section explores the choices of the lifting steps $l_i$ that result in high-performance transforms.
3.1. Approximating the DCT's rotation angles

The simplest method to construct a multiplierless approximation of the DCT is to approximate each of its rotation angles using cascades of dyadic lifting steps. We can first start with the fast 1D 8 × 8 DCT algorithm which can compute 8 coefficients with as little as 9 multiplications and 29 additions (13 multiplications are needed for correct scaling of all coefficients) [2]. Besides 8 butterfly pairs, this factorization consists of 5 rotation angles \( \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{3\pi}{4}, \frac{5\pi}{6} \) as shown in Figure 1. It is well-known that a plane rotation can be performed by 3 shears \([4]\), i.e.,

\[
\begin{bmatrix}
cos \theta_i & -\sin \theta_i \\
n \sin \theta_i & \cos \theta_i
\end{bmatrix} = \begin{bmatrix} 1 & u_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ p_i & 1 \end{bmatrix} \begin{bmatrix} 1 & u_i \\ 0 & 1 \end{bmatrix},
\]

where \( u_i = \frac{\cos \theta_i}{\sin \theta_i} \) and \( p_i = \sin \theta_i \). Hence, one can certainly try to approximate \( u_i \) and \( p_i \) by dyadic rationals so that binary arithmetic can be exploited to obtain fast, VLSI-friendly implementation. This leads to version A of the binDCT (labeled binDCT-A) with the following choices of lifting coefficients in the symmetric branches

1. Rotation \( \frac{\pi}{4} \approx \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -1 \\ 0 & 1 \end{bmatrix} \).

2. Rotation \( \frac{3\pi}{4} \approx \begin{bmatrix} 1 & 0 \\ -\frac{3}{8} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -\frac{3}{8} & 1 \\ 0 & 1 \end{bmatrix} \).

and the following approximations in the asymmetric branches

1. Rotation \( \frac{\pi}{4} \approx \begin{bmatrix} -1 & 0 \\ \frac{1}{2} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -1 \\ 0 & 1 \end{bmatrix} \).

2. Rotation \( \frac{3\pi}{4} \approx \begin{bmatrix} 1 & 0 \\ -\frac{12}{16} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -\frac{12}{16} & 1 \\ 0 & 1 \end{bmatrix} \).

3. Rotation \( \frac{5\pi}{16} \approx \begin{bmatrix} 1 & 0 \\ -\frac{1}{4} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ -\frac{1}{4} & 1 \\ 0 & 1 \end{bmatrix} \).

The ±1 butterflies are retained since their outputs are integers. Only the five rotation angles with floating-point multiplications are replaced. The binDCT-A approximates the DCT very closely. In fact, a cascade of the forward binDCT-A followed by the inverse floating-point DCT (or the forward original DCT followed by the inverse binDCT-A) yields a reasonable near-perfect-reconstruction system. The fast algorithm for the forward binDCT-A is shown in Figure 5.

3.2. General Low-Complexity Solutions

The binDCT-A described in the previous section is not the most efficient solution since the lifting steps are related. The computational complexity is not minimal due to the dependency between the lifting coefficients. We can limit the parameterization of each rotation angle to no more than two dyadic lifting steps. The reader should note that it is not necessary to approximate each rotation angle independently. Indeed, we have to take full advantage of their relationship to keep the number of lifting steps to a minimum.

For the \( U \) matrix which yields the symmetric basis functions, the following lifting choices yield a transform with high coding gain:

1. Rotation \( \frac{\pi}{4} \approx \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -1 \\ 0 & 1 \end{bmatrix} \).

2. Rotation \( \frac{3\pi}{4} \approx \begin{bmatrix} 1 & 0 \\ \frac{3}{4} & 1 \\ 0 & 1 \end{bmatrix} \).

Note that the refactorization of the lifting steps

\[
\begin{bmatrix} 1 & 0 \\ -\frac{3}{8} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{3}{8} & 1 \\ 0 & 1 \end{bmatrix}
\]

from the previous section into

\[
\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{3}{8} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{3}{8} \\ 0 & 1 \end{bmatrix}
\]

allows the elimination of the trailing permutation matrix by switching the index (order) of the two coefficients \( X[6] \) and \( X[2] \). Also, the \( \frac{\pi}{2} \) rotation can be approximated by any version of the unnormalized Haar. The choice of

\[
\begin{bmatrix} 1 & -\frac{1}{4} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{4} \\ 0 & 1 \end{bmatrix}
\]

works well too.

The \( V \) matrix which yields the antisymmetric basis functions poses a greater challenge. Specifically, the replacement of the rotation angle \( \frac{\pi}{4} \) in \( V \) has to be chosen wisely since it precedes, and hence greatly influences, the remaining lifting coefficients in the lattice. For example, the unnormalized Haar transform mentioned above does not yield very good result here. We propose the following choices:

1. Rotation \( \frac{\pi}{4} \approx \begin{bmatrix} -1 & \frac{\pi}{8} \\ 0 & 1 \end{bmatrix} \).

2. Rotation \( \frac{3\pi}{4} \approx \begin{bmatrix} 1 & -\frac{3}{8} \\ 0 & 1 \end{bmatrix} \).

3. Rotation \( \frac{5\pi}{16} \approx \begin{bmatrix} 1 & 0 \\ -\frac{12}{16} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ -\frac{12}{16} & 1 \\ 0 & 1 \end{bmatrix} \).

The resulting transform labeled binDCT-B is depicted in Figure 5; only the forward transform is shown. This binDCT version achieves 8.77 dB coding gain for AR(1) process (the DCT has 8.83 dB coding gain). It can compute 8 transform coefficients in 14 shift and 31 add operations. The transform becomes even nicer if we replace the \( \frac{3\pi}{16} \) rotation angle above in \( V \) by only two lifting steps:

\[
\begin{bmatrix} 1 & 0 \\ -\frac{1}{4} & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ -\frac{1}{4} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{4} & 1 \end{bmatrix}
\]

Although we cannot save any operation from this modification, the result labeled the binDCT-C is mathematically elegant since its transform matrix contains dyadic rationals of very limited dynamic range. The coding gain is still 8.77 dB; the complexity also stays the same – 14 shifts and 31 additions. The forward and inverse transform are depicted in Figure 6. The frequency responses of the binDCT-C are shown in Figure 8 and the filter coefficients are tabulated in Table 1.
Figure 5. High-performance binDCT versions. Left: binDCT-A. Right: binDCT-B.

Figure 6. The binDCT-C.

Figure 7. The all-lifting binDCT-C.
There is one problem with the binDCT versions presented in the previous sections: the dynamic range of the transform coefficients can be large (due to the unnormalized butterflies) making lossless compression infeasible. In order to solve this problem, we have to construct transforms consisting of lifting steps only. This is achieved quite easily by replacing every butterfly in the lattice by the unnormalized Haar. An example of such all-lifting transform is illustrated in Figure 7. To prevent excessive attenuation of any one subband (since the balance of the butterflies is sacrificed here), half of the butterflies are replaced by

$$\begin{bmatrix}
1 & 0 \\
\frac{1}{2} & -1
\end{bmatrix}\begin{bmatrix}
1 & 1 \\
0 & 1
\end{bmatrix} = \begin{bmatrix}
\frac{1}{2} & 1 \\
-\frac{1}{2} & -\frac{1}{2}
\end{bmatrix}$$

while the remaining half employs

$$\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix} = \begin{bmatrix}
\frac{1}{2} & 0 \\
\frac{1}{2} & -\frac{1}{2}
\end{bmatrix}.$$
Table 2. Objective coding result comparison (PSNR in dB).

<table>
<thead>
<tr>
<th>Comp. Ratio</th>
<th>Lena</th>
<th>Goldhill</th>
<th>Barbar</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DCT</td>
<td>binDCT-C</td>
<td>binDCT-B</td>
<td>DCT</td>
</tr>
<tr>
<td>Lossless</td>
<td>-</td>
<td>4.540 bpp</td>
<td>4.522 bpp</td>
<td>-</td>
</tr>
<tr>
<td>1:8</td>
<td>39.88</td>
<td>39.04</td>
<td>39.21</td>
<td>36.22</td>
</tr>
<tr>
<td>1:16</td>
<td>36.31</td>
<td>35.66</td>
<td>35.78</td>
<td>32.73</td>
</tr>
<tr>
<td>1:32</td>
<td>32.79</td>
<td>32.05</td>
<td>32.15</td>
<td>29.98</td>
</tr>
<tr>
<td>1:64</td>
<td>29.46</td>
<td>29.07</td>
<td>29.12</td>
<td>27.81</td>
</tr>
</tbody>
</table>

Figure 9. Enlarged 256 x 256 portions of the original and the reconstructed Barbara images at 1:16 compression ratio. From left to right: floating-point DCT, 31.11 dB; binDCT version B, 30.67 dB; binDCT version C, 30.19 dB.

- The binDCT has a fast, elegant implementation utilizing only shift-and-addr operations. No multiplication is needed. Eight transform coefficients can be computed using only 14 bit shifts and 31 additions (13 floating-point multiplications and 29 additions are required for the DCT).
- The binDCT can map integers to integers with exact reconstruction. This property is pivotal in transform-based lossless coding. It also allows a unifying lossy/lossless coding framework.
- In software implementation, the binDCT is already 3–4 times faster than the floating-point DCT. Much better saving in speed is expected in hardware implementation.
- The multiplierless property of the binDCT allows efficient VLSI implementations in terms of both chip area and power consumption.
- The binDCT approximates the DCT very closely. Perceptual quantization matrices and coding strategies designed specifically for the DCT can be applied to the binDCT immediately without any complicated modification.
- The cascade of a well-tuned forward binDCT followed by an inverse DCT (or a forward DCT followed by an inverse binDCT) produces a reasonable near-perfect-reconstruction system.

The binDCT provides reasonably high coding performances. The coding gain of several binDCT versions ranges from 8.77 to 8.82 dB (the DCT has 8.83 dB coding gain). In our image coding experiments, the binDCT is around 0.1 - 1.0 dB below the DCT in the PSNRs of the reconstructed images. Generalizing the concept in this paper to longer filter lengths (resulting in overlapping block transforms) is also relatively straightforward [7].

REFERENCES